The Impact of User Feedback on Drug Success

A Data-Driven Approach to Analyzing Ratings and Conditions

Ali Mohtat, Omar Sagoo

Abstract

This project explores the application of machine learning techniques to analyze user-generated reviews of medications, providing insights into drug effectiveness, patient sentiment, and the prevalence of side effects. Utilizing the Drug Review Dataset from Drugs.com, which includes over 213,000 patient reviews, this study employs sentiment analysis, side effect detection, and classification models to uncover patterns in user feedback across a wide range of medical conditions. In addition to the data analysis, a graphical interface was developed to assist healthcare professionals in identifying the most effective drugs for specific conditions based on aggregated patient feedback. The interface presents drug recommendations along with confidence scores, offering a user-friendly tool for informed decision-making. The results of this project highlight the potential of leveraging large-scale user data to complement clinical research, enhance treatment decisions, and improve patient care.

Introduction

In the medical field, data-driven insights have become crucial for improving healthcare outcomes and making informed decisions. Traditional methods, such as clinical trials and structured surveys, have long been the cornerstone of gathering reliable data on drug effectiveness, patient satisfaction, and potential side effects. These methods, however, are often limited by controlled conditions and relatively small sample sizes, which may not fully reflect the diversity of real-world patient experiences.

As healthcare continues to evolve, new sources of data have emerged that complement clinical trials, offering a broader and more nuanced understanding of medication performance. One such source is user-generated content, where patients share their personal experiences with medications on online platforms. These reviews provide valuable real-world insights into drug efficacy and adverse effects, often covering a wider range of conditions and patient demographics than traditional studies.

This project leverages the Drug Review Dataset from Drugs.com, which is publicly available through the UC Irvine Machine Learning Repository. The dataset comprises over 213,000 user

reviews, providing a rich resource for analyzing patient sentiments on drug effectiveness and side effects across various medical conditions.

By conducting sentiment analysis and clustering techniques on this dataset, we aim to uncover patterns in user feedback, helping to inform both healthcare professionals and patients. Our analysis builds on existing research, such as Gräßer et al. (2018), which explored cross-domain learning in drug reviews, demonstrating the potential for deriving insights from large-scale user data. This project aims to further these efforts by evaluating drug effectiveness and user perceptions in a comprehensive, data-driven manner.

Data Cleaning/Preparation

The Drug Review Dataset from Drugs.com, available through the UC Irvine Machine Learning Repository, contains user-generated reviews of various medications. The dataset comprises 6 variables:

- **drugName**: The name of the drug being reviewed (categorical).
- **condition**: The medical condition for which the drug was prescribed (categorical).
- **review**: The text of the review provided by the user (text).
- **rating**: A numerical rating provided by the user on a 10-point scale (numeric).
- **date**: The date the review was submitted (date).
- **usefulCount**: The number of users who found the review helpful (numeric).

The dataset contains 213,869 entries, providing a substantial amount of feedback on various drugs and conditions. The data cleaning process focused on ensuring completeness and summarizing key variables, as detailed below.

1. **Missing Value Treatment**:

   An examination of the dataset revealed that 1,194 entries in the condition column were missing. These rows were removed to maintain the accuracy of the analysis. Other key variables, such as drugName, review, rating, date, and usefulCount, were complete and did not require further handling of missing values.

2. **Improper Condition Value:**

   Further examination of the dataset revealed that 1,171 rows included improper condition values. The values of the row in the condition column would be “92 users found this review helpful”. Since this is not labeled with an actual condition, the data should be removed as to not pollute the dataset.

3. **Numerical Summary**:

   Summary statistics were generated for the numerical variables, rating and usefulCount. These statistics provided insight into the distribution of user feedback:

   - The average user rating was 6.99, with ratings ranging from 1 to 10.
   - The number of useful votes on reviews ranged from 0 to 1,291, with an average of 28.

4. **Categorical Variables Overview**:

   The dataset contains 3,667 unique drug names and 916 unique medical conditions. These

categorical variables were not transformed but were counted and summarized to understand the variety of drugs and conditions reviewed.

5. **Textual Review Summary**:

   The length of user reviews was analyzed by calculating the number of words in each review. On average, reviews contained 85 words. This information provided a foundation for sentiment analysis and future text-based modeling.

By addressing missing values and summarizing key numerical and categorical variables, the dataset was prepared for further analysis. The next stage of the project involved exploratory data analysis and modeling, building on these foundational steps.

Exploratory Data Analysis (EDA)

The exploratory data analysis (EDA) involved examining the distribution of ratings, usefulness of reviews, correlations between key features, and the temporal trends in both ratings and reviews. These analyses provide insights into user perceptions of drugs, rating behavior, and changes in feedback over time. Key findings and their interpretations are detailed below.

1. Distribution of Ratings and Usefulness

The numerical columns rating and usefulCount were explored to understand how users rate drugs and how helpful other users find these reviews.

- **Histogram and Density Plot**: The distribution of ratings (Figure 1) shows that users tend to give either very high or very low ratings, with a pronounced spike at 10. This suggests that users are more inclined to express strong opinions, either positively or negatively.

Such a distribution indicates that users may only leave reviews when they have extreme experiences (either highly satisfied or dissatisfied) with a medication.

- **Box Plot**: The box plot further confirms the skewed nature of the ratings, with the median rating around 8 and a significant spread in the upper quartile. Outliers in the lower range suggest a small but notable group of dissatisfied users.
- **Useful Count Distribution**: The distribution of useful counts (Figure 1) shows that most reviews received only a few useful votes, with a few exceptions where reviews were marked as useful by many users. This pattern could indicate that users gravitate toward highly informative or particularly detailed reviews when looking for useful content.

The rating data indicates a general trend of high satisfaction among users, but it also highlights the polarization in user opinions. Most users seem to either love or hate a medication, which is important when considering the average rating of a drug as a meaningful indicator of user satisfaction.

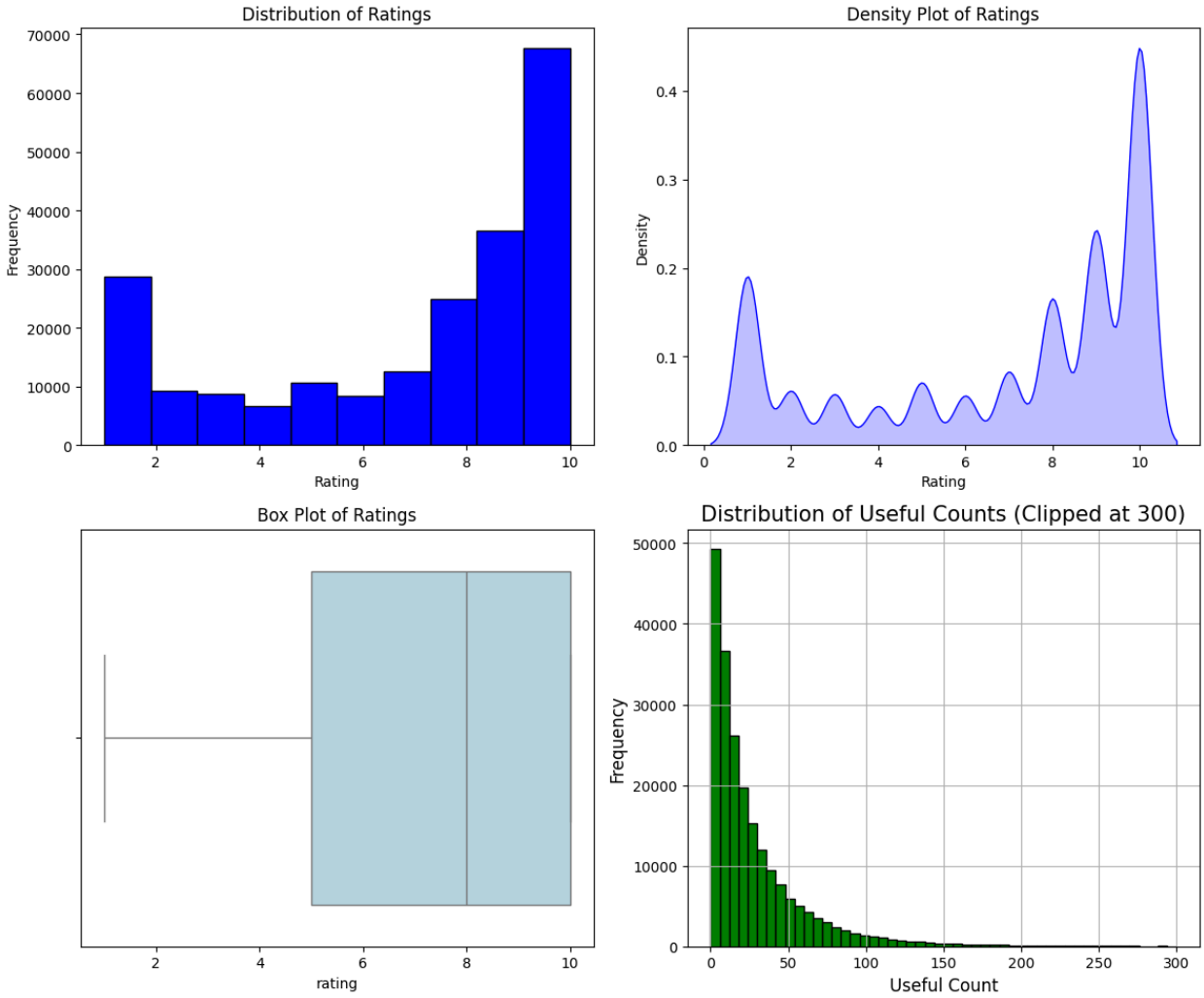


Figure 1 Distribution of Ratings and Usefulness

2. Average Ratings per Condition

Analyzing the average ratings per medical condition provides insights into how well drugs are perceived for specific conditions.

- **Top 10 Conditions with Highest Ratings**: Conditions like Smoking Cessation, Opiate Dependence, and Emergency Contraception received the highest ratings (Figure 2). These high ratings suggest that medications for these conditions are perceived to be effective by users, potentially because they address highly specific and impactful medical needs.

- **Top 10 Conditions with Lowest Ratings**: Conversely, conditions like Insomnia, Major Depressive Disorder, and High Blood Pressure have the lowest average ratings (Figure 2). This could be due to the chronic nature of these conditions, where patients may not experience the desired level of improvement, or where side effects of medications are more prominent.

The disparity in ratings between conditions highlights the complexity of treating chronic versus acute conditions. For conditions where immediate relief is common (e.g., smoking cessation), users tend to be more satisfied, whereas chronic conditions often have lower satisfaction due to the complexity of treatment and potential side effects.

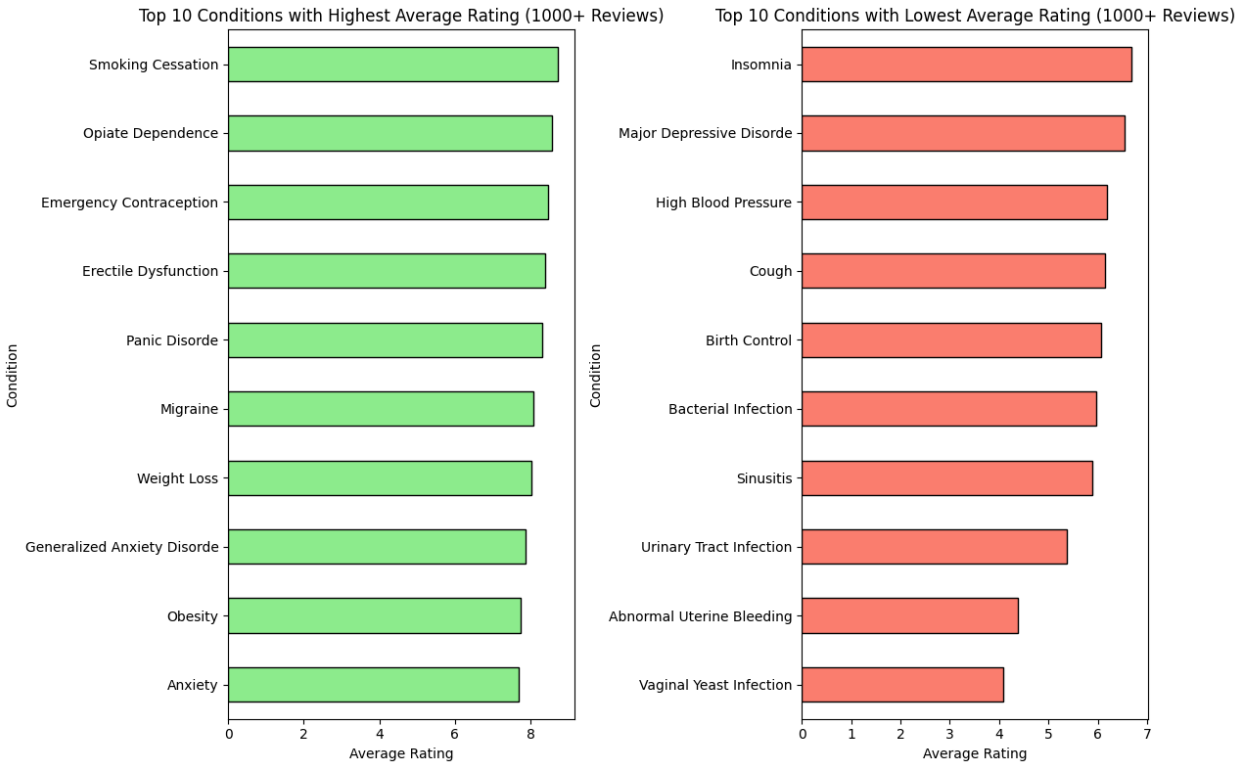


Figure 2 Average Ratings per Condition

3. Correlation Analysis of Numerical Features

A correlation matrix was generated to explore relationships between key numerical features. Correlation values range from -1 (strong negative correlation) to 1 (strong positive correlation). Figure 3 presents a heatmap displaying these correlations.

- **Key Insights**:
    - A moderate positive correlation exists between review_length and side_effects (0.30), indicating that longer reviews tend to mention side effects more frequently.
    - rating and usefulCount have a moderate positive correlation (0.24), implying that reviews with higher ratings often receive more helpful votes.
    - side_effects and rating exhibit a slight negative correlation (-0.095), suggesting that reviews mentioning side effects tend to give lower ratings.

This analysis suggests that users who experience side effects are more likely to write longer reviews and provide lower ratings. Additionally, highly rated reviews are often more helpful to other users, leading to more helpful votes.

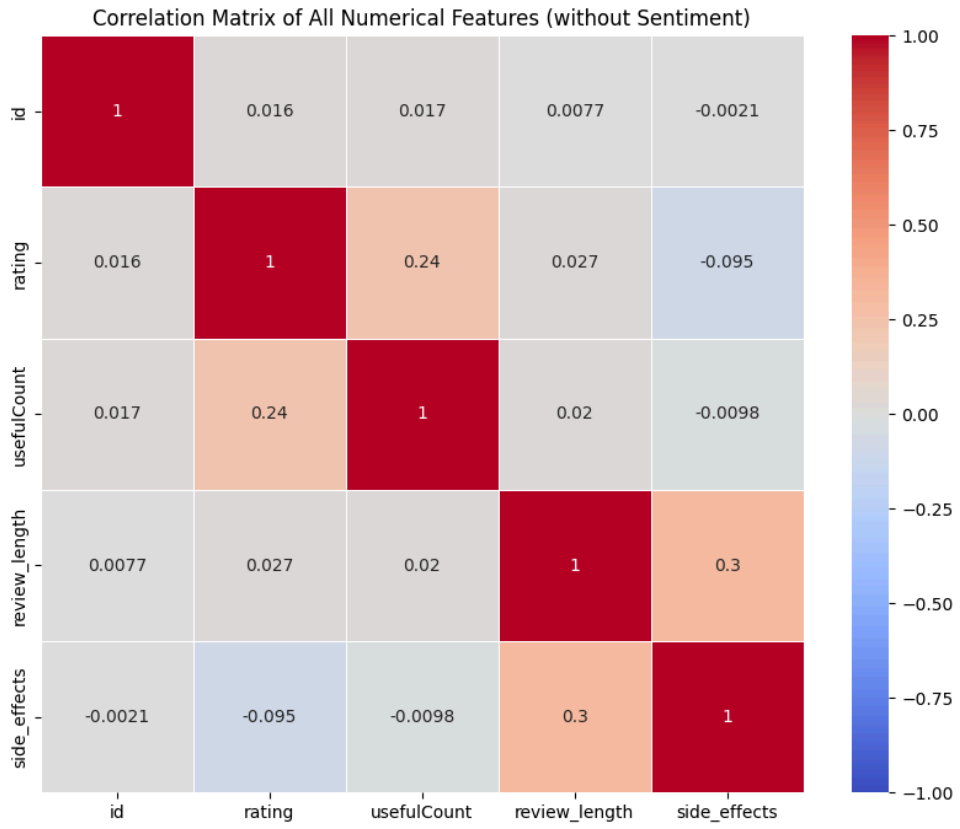


Figure 3 Heatmap of the correlation matrix

4. Temporal Analysis of Reviews and Ratings

Temporal trends were analyzed to investigate whether user ratings change over time or across different months.

- **Temporal Trends by Year**: The temporal analysis of the top 10 most-reviewed drugs and conditions shows fluctuations in ratings over time (Figure 4). Some drugs and conditions experienced consistent ratings, while others, such as those for mental health conditions, saw a gradual decline in ratings over the years. This decline might reflect user disappointment as more patients share their experiences, or it could indicate changing expectations or the introduction of competing medications.

- **Temporal Trends by Month**: The month-to-month analysis of ratings reveals that while there are some fluctuations, no significant seasonal patterns are detected (Figure 4). However, slight variations in ratings during certain months may indicate external factors such as healthcare trends or media coverage affecting user perceptions.

The temporal decline in average ratings for some drugs could signal either drug inefficacy becoming more apparent over time or new drugs entering the market and shifting user preferences. On the other hand, the consistency in ratings for other drugs suggests that these medications have maintained their perceived effectiveness over the years.

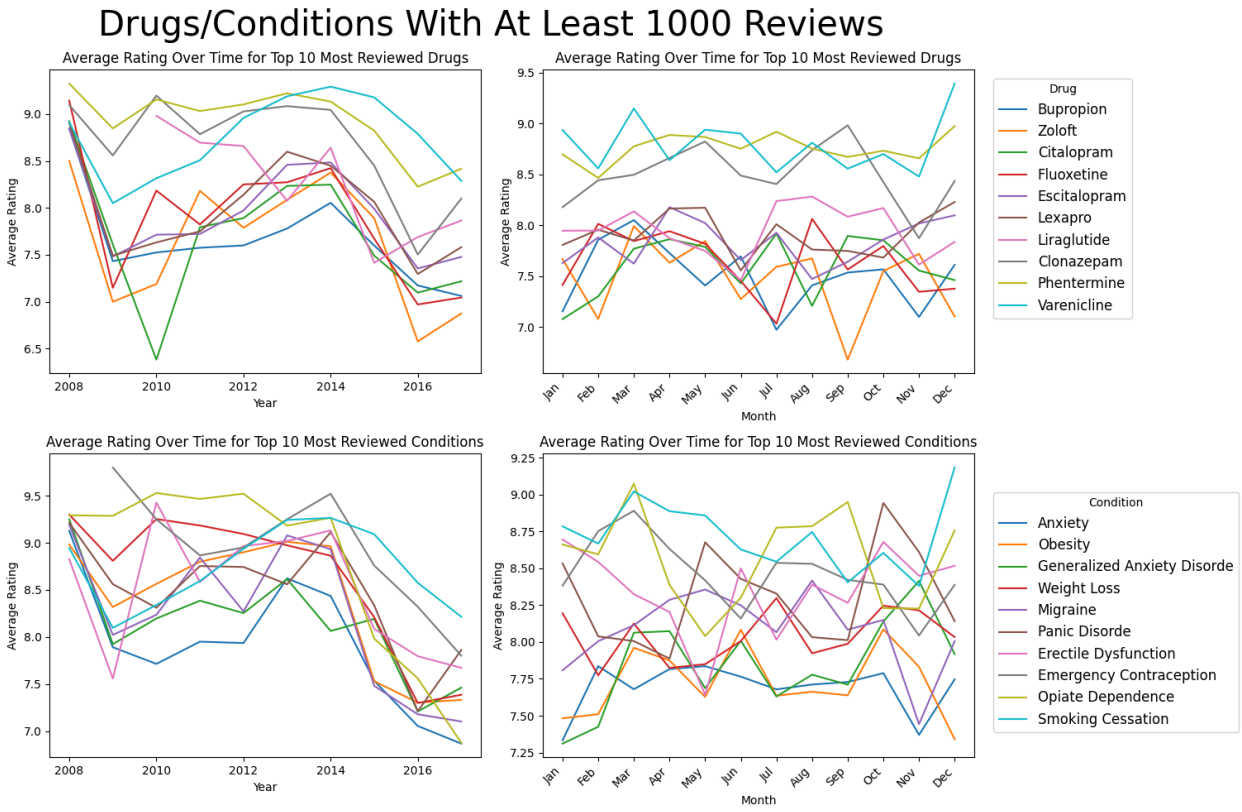


Figure 4 Temporal Analysis of Reviews and Ratings

5. Top Reviewed Drugs and Conditions

To further explore the dataset, the top 10 most-reviewed drugs and conditions were identified and visualized (Figure 5 and Figure 6). Levonorgestrel and Birth Control were the most commonly reviewed drug and condition, respectively.

- **Top Drugs**: Levonorgestrel was the most-reviewed drug with nearly 5,000 reviews, followed by Etonogestrel and Ethinyl estradiol/norethindrone.
- **Top Conditions**: Birth Control dominated the condition reviews, with over 38,000 reviews, followed by Depression and Pain.

This analysis highlights the significant interest in birth control-related medications and conditions, likely reflecting the widespread use and personal importance of these treatments.

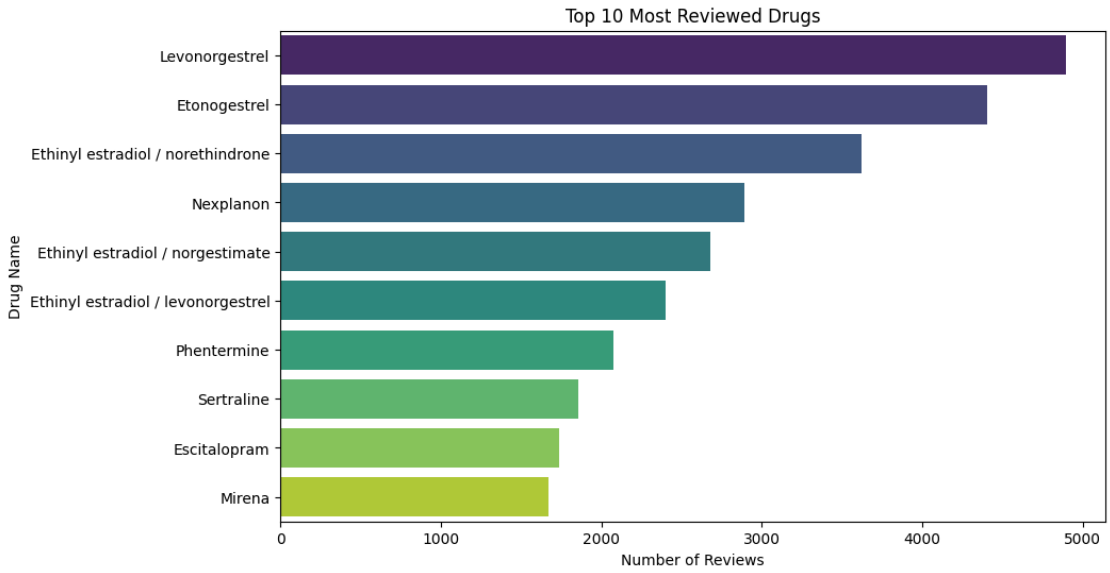


Figure 5 Top 10 most-reviewed drugs

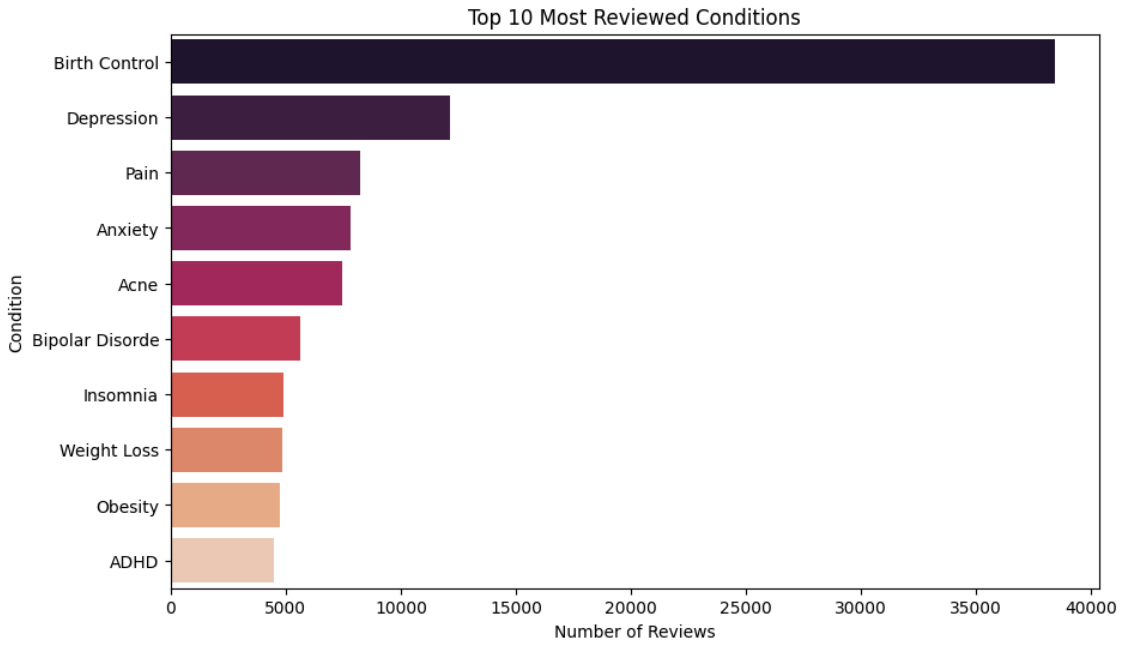


Figure 6 Top 10 most-reviewed conditions

6. Temporal Trends in Review Volume and Ratings

The relationship between the number of reviews and average ratings was explored to see if the volume of feedback affects user ratings.

- **Temporal Trends in Review Volume and Ratings**: The trend line (Figure 7) shows that as the number of reviews increased, particularly after 2012, the average ratings began to decline. This pattern could indicate that as more users review a drug, a broader range of opinions (both positive and negative) are represented, leading to lower overall ratings. This trend suggests that early adopters of a drug may have been more satisfied, while later users may have had different expectations or experiences.

The inverse relationship between the number of reviews and the average rating suggests that new drugs might initially receive positive feedback, but as more users provide reviews, the ratings

may become more moderate or negative. This could reflect a broader range of user experiences and potentially changing user expectations.

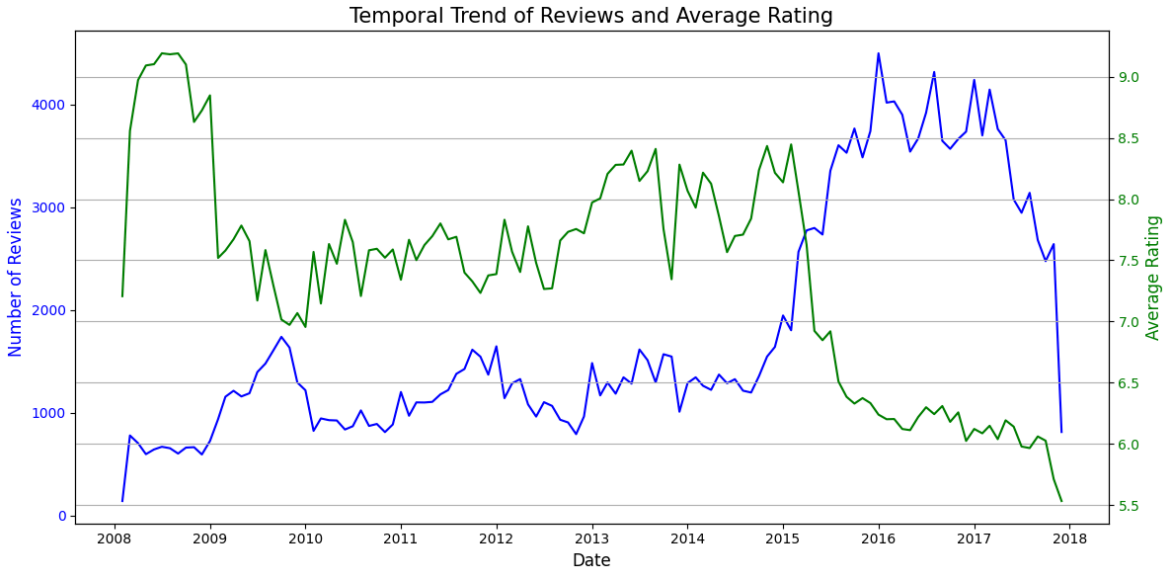


Figure 7 Review Volume and Ratings as function of time

7. Focused Analysis: ADHD Condition

In addition to the general analysis, a more focused examination was performed on the ADHD condition, which represents a significant portion of the reviews in the dataset. This sub-analysis aims to provide deeper insights into user satisfaction and the effectiveness of drugs specifically used to treat ADHD.

- **Distribution of Ratings for ADHD Drugs**: A similar pattern of skewed ratings was observed for ADHD drugs, with most ratings falling between 8 and 10, indicating high levels of satisfaction among users. However, there are still notable outliers with lower ratings, suggesting some dissatisfaction for a subset of users.

- **Average Ratings per Drug for ADHD**: ADHD drugs were ranked based on their average ratings (Figure 8). Dextrostat and Adderall XR emerged as the top-rated drugs, reflecting strong user satisfaction with these treatments. The ratings for these drugs are consistently high, suggesting their perceived effectiveness among patients dealing with ADHD symptoms.

- **Word Cloud of ADHD Drugs**: A word cloud was generated to visualize the most frequently reviewed ADHD drugs (Figure 8). Drugs like Adderall XR and Vyvanse are prominently featured, which highlights their widespread use and the attention they receive from the ADHD community. These drugs have a reputation for being highly effective in managing symptoms, as evidenced by the high volume of reviews and ratings.

The focused analysis of ADHD medications reveals that despite a wide array of treatments, certain drugs such as Adderall XR and Vyvanse dominate both in terms of usage and user satisfaction. This could suggest that these medications have established themselves as reliable options for managing ADHD symptoms, leading to a larger pool of user feedback and consistently high ratings.

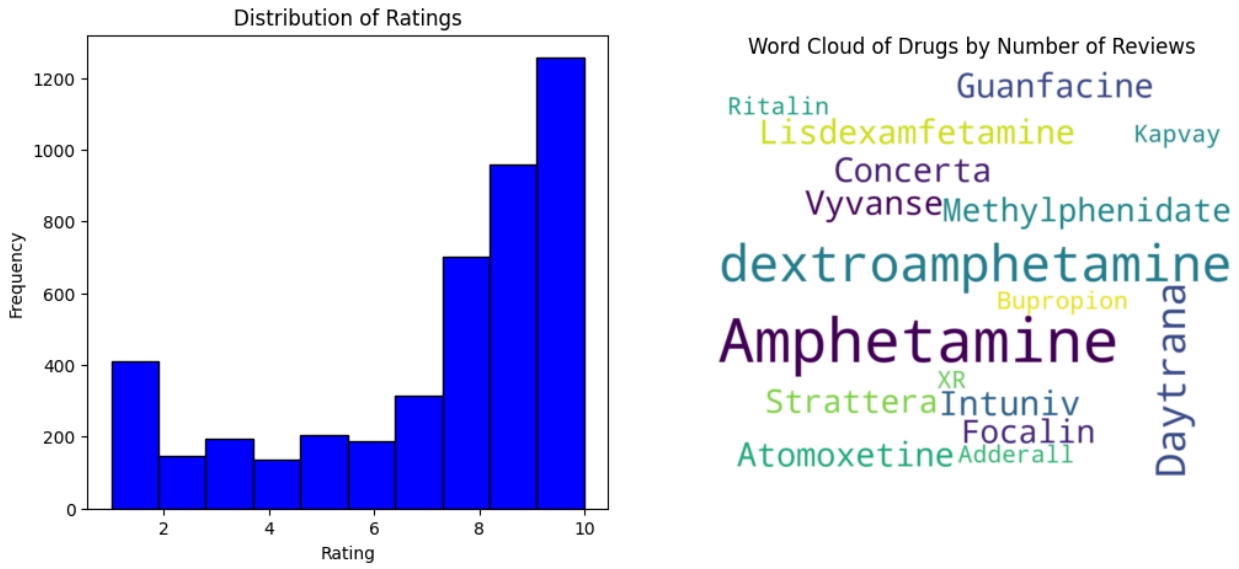


Figure 8 Distribution of ratings and word cloud for ADHD condition

Model Selection

In this analysis, several techniques were used to predict the effectiveness of drugs and detect the side effects mentioned in user reviews. The models chosen focus on different aspects of the data: identifying side effects from user reviews, understanding the sentiment behind the reviews, and using machine learning to predict whether a drug will be effective based on its characteristics. Below is an explanation of each approach, along with the results and interpretation of the findings.

1. Side Effect Detection

Side effects are a critical consideration for patients and healthcare providers when evaluating the use of medications. In this model, we built a custom method to identify whether a review mentions side effects. The system scans the text for specific keywords related to side effects

(e.g., "headache," "nausea," "dizziness") and checks if these mentions are negated (e.g., "no side effects").

- **Method**: The system relies on a list of commonly reported side effects and searches user reviews for these terms. If a side effect is mentioned and is not negated by words like "no" or "not," the system flags the review as containing a side effect.
- **Result**: Figure 9 shows the top 10 drugs with the most reviews mentioning side effects. Medications like Levonorgestrel and Etonogestrel (both hormonal contraceptives) had the highest number of reviews reporting side effects. This aligns with known side effects for hormonal treatments, which can include symptoms like headaches, nausea, and mood changes.

The results suggest that users of certain contraceptive drugs are more likely to mention side effects in their reviews. These findings could help patients and healthcare providers understand which medications might be more prone to adverse effects, guiding decision-making when considering treatment options.

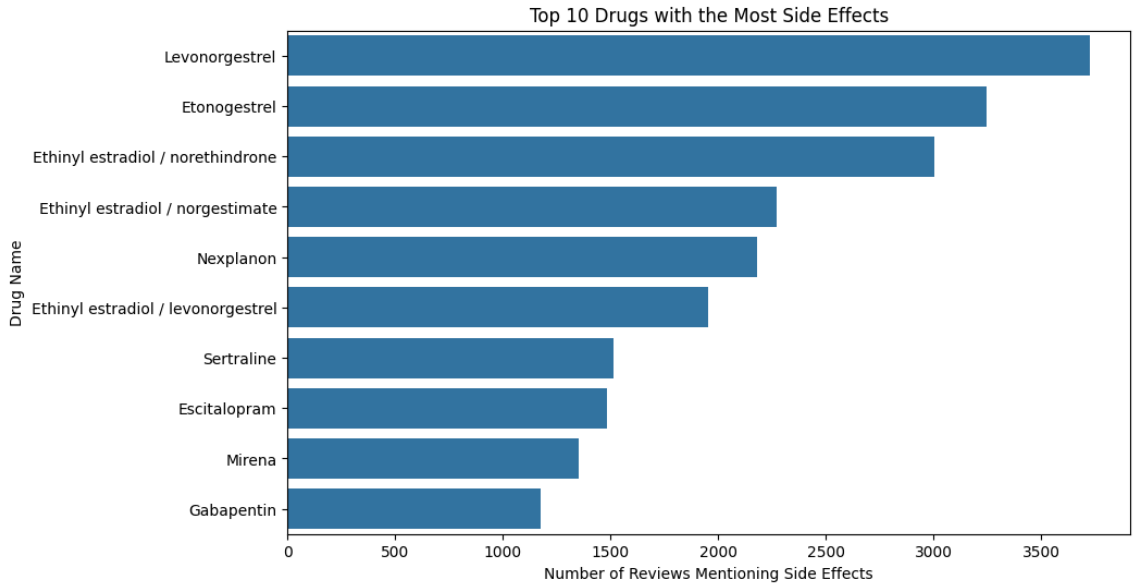


Figure 9 Top 10 drugs with most side effects.

2. Sentiment Analysis and Rating Distribution

Sentiment analysis is a technique that determines the emotional tone of a piece of text. In this case, we used sentiment analysis to assess whether user reviews were generally positive, negative, or neutral. The tool used for this analysis, VADER (Valence Aware Dictionary and sEntiment Reasoner), is specifically designed to handle social media text but works well with short user reviews too. VADER assigns a sentiment score to each review, with negative scores indicating negative sentiment, and positive scores indicating positive or neutral sentiment (Hutto & Gilbert, 2014).

- **Method**: The sentiment score for each review was computed, and the reviews were grouped into two categories: negative sentiment (sentiment score < 0) and neutral/positive sentiment (sentiment score ≥ 0). We then compared the distribution of ratings between these two groups to see how user sentiment correlates with their numerical rating of the drug.

- **Result**: Figure 10 shows that reviews with negative sentiment tend to give lower ratings (around 1-2), while neutral and positive reviews generally have much higher ratings (7-10). This strong correlation between sentiment and rating indicates that sentiment analysis is a good predictor of how satisfied users are with a drug.

Users who express negative emotions in their reviews tend to give low ratings, while those with positive or neutral sentiment rate the drugs highly. This suggests that sentiment analysis can serve as an additional layer of insight into user satisfaction and could be useful for future analysis or recommendations.

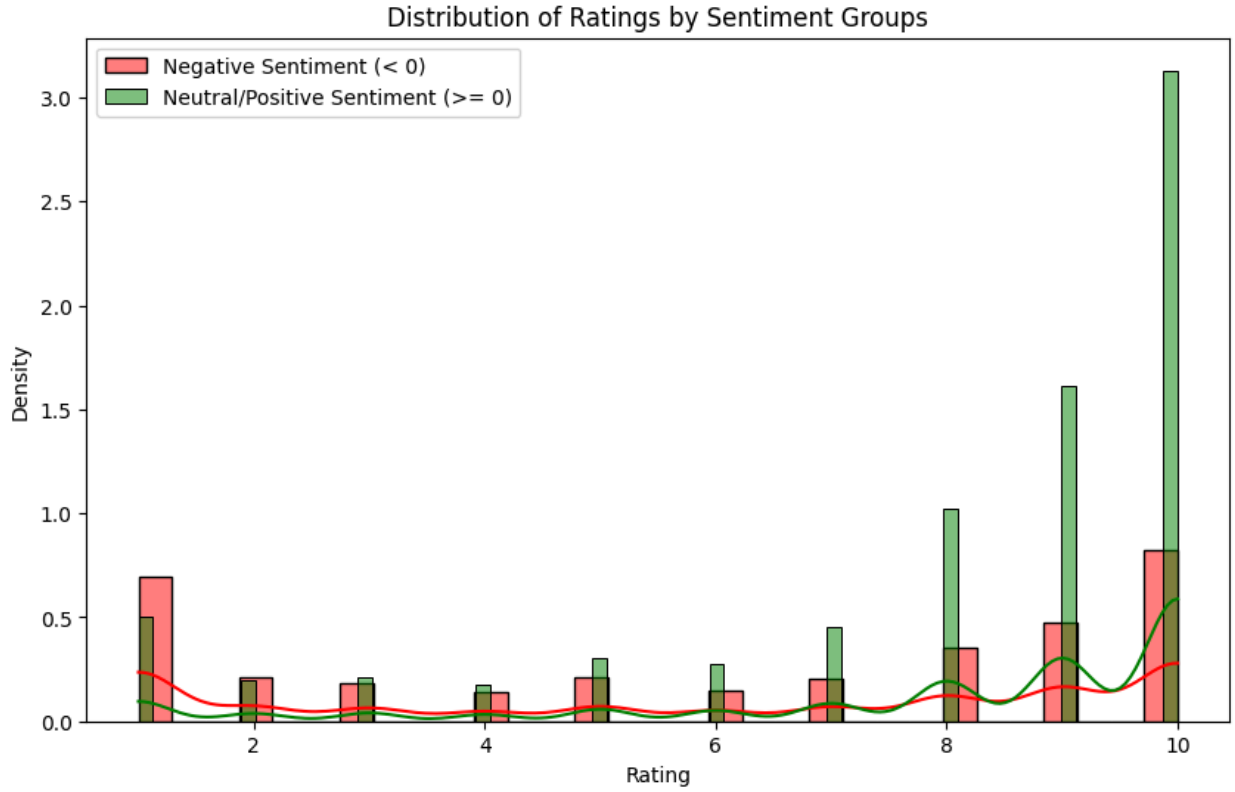


Figure 10 *Distribution of ratings by sentiment groups.*

3. Random Forest Classifier for Predicting Drug Effectiveness

The Random Forest model is a widely used machine learning algorithm that excels at making predictions based on complex data. It operates by building a "forest" of decision trees, each trained on different parts of the data, and then aggregating their results to make a final prediction. This method is particularly useful when dealing with data that contains multiple variables, as it can capture intricate relationships between them.

- **Background**: The Random Forest model was chosen because it handles both categorical and numerical data well, and it is less likely to overfit compared to simpler models. In our case, the model was tasked with predicting whether a drug would be effective (defined as a rating > 5) based on the drug's name and the medical condition it treats.
- **Method**: The dataset was split into two parts: 75% for training the model and 25% for testing. The drug name and condition were converted into numerical values (since machine learning models work better with numbers). The model learned to associate certain drug-condition pairs with effectiveness based on the training data and was then tested on the remaining data to evaluate its performance.
- **Result**: The model achieved a training accuracy of 73.25% and a testing accuracy of 70.95%. This means that about 71% of the time, the model correctly predicted whether a drug would be effective based on the drug name and the condition. The classification report shows that the model is much better at predicting effectiveness (class 1) than non-effectiveness (class 0), with high precision and recall for effective drugs.

The Random Forest model successfully predicted drug effectiveness with reasonable accuracy. However, the model struggles to predict when a drug will not be effective, which could be due to

the class imbalance (there are more reviews of effective drugs than non-effective ones). This could be addressed in future iterations of the model by adding more features (e.g., user reviews, dosage) or by balancing the dataset.

To further evaluate the Random Forest model, a comparison between the actual effectiveness (based on user ratings) and the predicted effectiveness (from the model) was plotted (Figure 11). This comparison helps visualize how well the model performs and where it might need improvement. The plot shows that the model correctly predicted effectiveness in the majority of cases. However, it struggled to correctly predict non-effectiveness (class 0), as the model tends to classify most drugs as effective.

The Random Forest model performs well in predicting when a drug will be effective but has difficulty predicting when it will not be effective. This might be due to the limited number of reviews for non-effective drugs. Improvements such as addressing class imbalance or incorporating additional variables (like review text or sentiment) could enhance the model's accuracy.

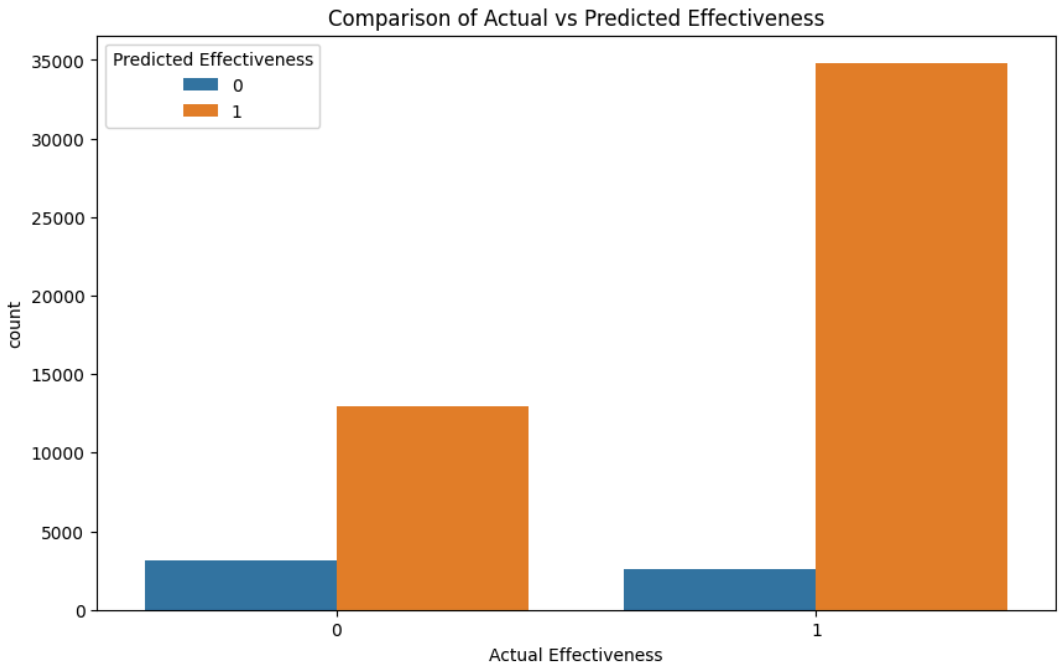


Figure 11 Comparison of actual vs. predicted drug effectiveness

Graphical Interface for Drug Effectiveness Prediction

In this project, a graphical interface was developed to allow healthcare providers to identify the most effective drugs for specific medical conditions based on patient reviews. The objective of this tool is to provide insights into drug effectiveness and assist medical staff in making informed decisions. By selecting a medical condition from a dropdown menu, the interface presents the best-rated drug for that condition along with a confidence score that reflects the consistency of patient-reported effectiveness.

Methodology and Assumptions

The methodology behind the graphical interface is structured to ensure accuracy and usability. Below, the steps involved in its development are outlined:

1. Defining Effectiveness Using Percentiles

The effectiveness of a drug is determined by analyzing patient ratings. Rather than relying solely on average ratings, the system computes percentile rankings. A percentile ranking provides a comparison of a drug's performance relative to others for the same condition. For example, a drug in the 90th percentile is rated higher than 90% of other drugs for the same condition.

This approach introduces more variance into the effectiveness measure, allowing for a more nuanced understanding of drug performance across various conditions.

2. Confidence Calculation

The confidence score is a measure of reliability, reflecting how consistently a drug is rated highly for a given condition. It is calculated by taking the mean percentile rating of the drug across all relevant reviews for a condition. A higher confidence score indicates that a drug consistently performs well across a large number of patient reviews.

For instance, a drug with a confidence score of 85% suggests that, based on patient feedback, this drug is consistently rated as more effective than 85% of other drugs for the same condition.

3. Selection of the Best Drug

When a medical provider selects a condition from the dropdown menu, the system searches through the reviews to identify the drugs associated with that condition. The drug with the highest confidence score is presented as the best option for that condition.

This ranking is based on the aggregation of user reviews and offers insights into drug effectiveness for the selected condition.

4. User Interface Implementation

The interface includes a dropdown menu that allows users to select from a list of medical conditions. Once a condition is selected, the system displays the best-rated drug along with its confidence score, reflecting the reliability of the recommendation.

The interface is designed to be user-friendly and requires no specialized knowledge of statistics or machine learning. The graphical interface offers several key benefits for healthcare providers, particularly in clinical settings where quick access to reliable information is crucial.

- **Rapid Access to Patient Feedback**: The interface provides immediate insights into which drugs are perceived as the most effective by patients, based on thousands of reviews. This enables healthcare providers to quickly identify treatment options without needing to manually analyze large datasets.
- **Confidence in Recommendations**: The inclusion of a confidence score offers healthcare providers an additional layer of certainty in their treatment decisions. A higher confidence score suggests that the drug is consistently effective across a wide range of patient reviews, increasing trust in the recommendation.
- **Data-Driven Decision-Making**: By utilizing data from a large, diverse population, healthcare providers can make evidence-based decisions, leading to more personalized and effective treatments. This approach is particularly useful when considering medications for conditions that may not have extensive clinical trial data.
- **Ease of Use**: The interface is designed to integrate seamlessly into healthcare workflows. With its simple, intuitive design, the tool can be used by medical professionals without any need for specialized training or technical expertise.

For example, if a healthcare provider is treating a patient for **ADHD**, they can select "ADHD" from the dropdown menu. The interface may then return **"Dextrostat (Confidence: 86.07%)"** as the recommended drug. This means that, based on patient reviews, Dextrostat has a high confidence score for treating ADHD, reflecting its consistent effectiveness based on user feedback.

Conclusion and Recommendations:

This project has demonstrated the potential of leveraging large-scale user-generated reviews to gain valuable insights into drug effectiveness, side effects, and overall patient satisfaction. By applying machine learning models and data analysis techniques to the Drug Review Dataset, we were able to assess the emotional tone of reviews through sentiment analysis, detect mentions of side effects, and predict drug effectiveness using classification models. The results highlight the utility of patient reviews as a supplementary source of information for healthcare decision making, particularly when aggregated at scale.

The Sentiment Analysis and Side Effect Detection provided a deeper understanding of the correlation between patient sentiments and drug ratings. For example, reviews with negative sentiment frequently corresponded with lower ratings, while drugs associated with positive sentiment typically received higher ratings. The custom-built model for identifying side effects further enabled the automatic detection of common side effects mentioned in patient reviews, giving healthcare providers an additional layer of insight into patient experiences.

The Random Forest Classifier used for predicting drug effectiveness achieved reasonable accuracy but indicated room for improvement, particularly in predicting non-effectiveness.

Future efforts to balance the dataset or incorporate additional features, such as dosage information or patient demographics, could enhance the model's performance.

The Graphical Interface provided an intuitive way for healthcare professionals to access drug information, simplifying the process of identifying the most effective medications for specific conditions. The tool's use of percentile-based confidence scores makes it easier for users to trust the recommendations, which are based on aggregated patient feedback rather than clinical trial data alone.

Recommendations:

1. **Expand the Dataset**: To improve model accuracy and applicability, we recommend expanding the dataset to include other patient feedback sources, such as social media or additional healthcare platforms. This would create a more comprehensive view of drug performance across different demographics and conditions.
2. **Address Class Imbalance**: Future iterations of the classification model should address the imbalance between reviews of effective and non-effective drugs. Techniques such as oversampling, undersampling, or using synthetic data generation (e.g., SMOTE) can help improve the model's predictive accuracy.
3. **Integrate Additional Features**: Incorporating additional variables such as dosage information, treatment duration, or patient demographics (e.g., age, gender) could enhance the predictive power of the models and provide more personalized insights into drug effectiveness.
4. **Further Develop the Interface**: The current graphical interface could be expanded to include additional features, such as comparison tools that allow healthcare providers to

compare multiple drugs side by side. This would offer greater flexibility in decision-making and support more detailed analyses.

In conclusion, this project demonstrates the power of combining machine learning techniques with patient-generated content to derive meaningful, actionable insights. By continuing to develop these methods, healthcare providers can benefit from more data-driven, patient-centered approaches to prescribing medications and improving treatment outcomes.

References

Gräßer, F., Kallumadi, S., Malberg, H., & Zaunseder, S. (2018). Aspect-based sentiment analysis of drug reviews applying cross-domain and cross-data learning. *Journal of Biomedical Informatics*, 84, 136–147.

Hutto, C., & Gilbert, E. (2014). VADER: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth International Conference on Weblogs and Social Media (ICWSM-14)*.

Appendix A: Drug Analysis Notebook (complete with outputs)

In [1]:
```
# Imports
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import seaborn as sns
import numpy as np
import pandas as pd
from wordcloud import WordCloud
from ucimlrepo import fetch_ucirepo

pd.options.mode.chained_assignment = None
```

In [2]:
```
# fetch dataset from the UCI archive website by its ID.
drug_dataset_id = 462
Drugs = fetch_ucirepo(id=drug_dataset_id)
```

In [3]:
```
# variable info.
print(Drugs.metadata["additional_info"]["variable_info"])
```

1. drugName (categorical): name of drug
2. condition (categorical): name of condition
3. review (text): patient review
4. rating (numerical): 10 star patient rating
5. date (date): date of review entry
6. usefulCount (numerical): number of users who found review useful

In [4]:
```
# Create a total data from the Drugs
data_original = Drugs.data.original

# Checking for missing values in the dataset
missing_values = data_original.isnull().sum()

# Drop the rows with the missing condition value.
data_original = data_original.dropna()

# Drop the rows with an improper condition. rows with </span> do not include
span_count = data_original['condition'].str.contains('</span>', na=False).su
data_original = data_original[~data_original['condition'].str.contains('</sp

# Basic summary statistics for numerical columns
numerical_summary = data_original.describe()

# Distribution of unique values for categorical columns
drug_count = data_original['drugName'].nunique()
condition_count = data_original['condition'].nunique()

# Summarize text review column (average length of reviews)
data_original['review_length'] = data_original['review'].apply(lambda x: ler

average_review_length = data_original['review_length'].mean()

# Display key stats
print(f"Missing Values \n{missing_values}\n")
print(f'Number of rows with "</span>": {span_count}')
print(f"Numerical Summary: \n{numerical_summary}\n")
```

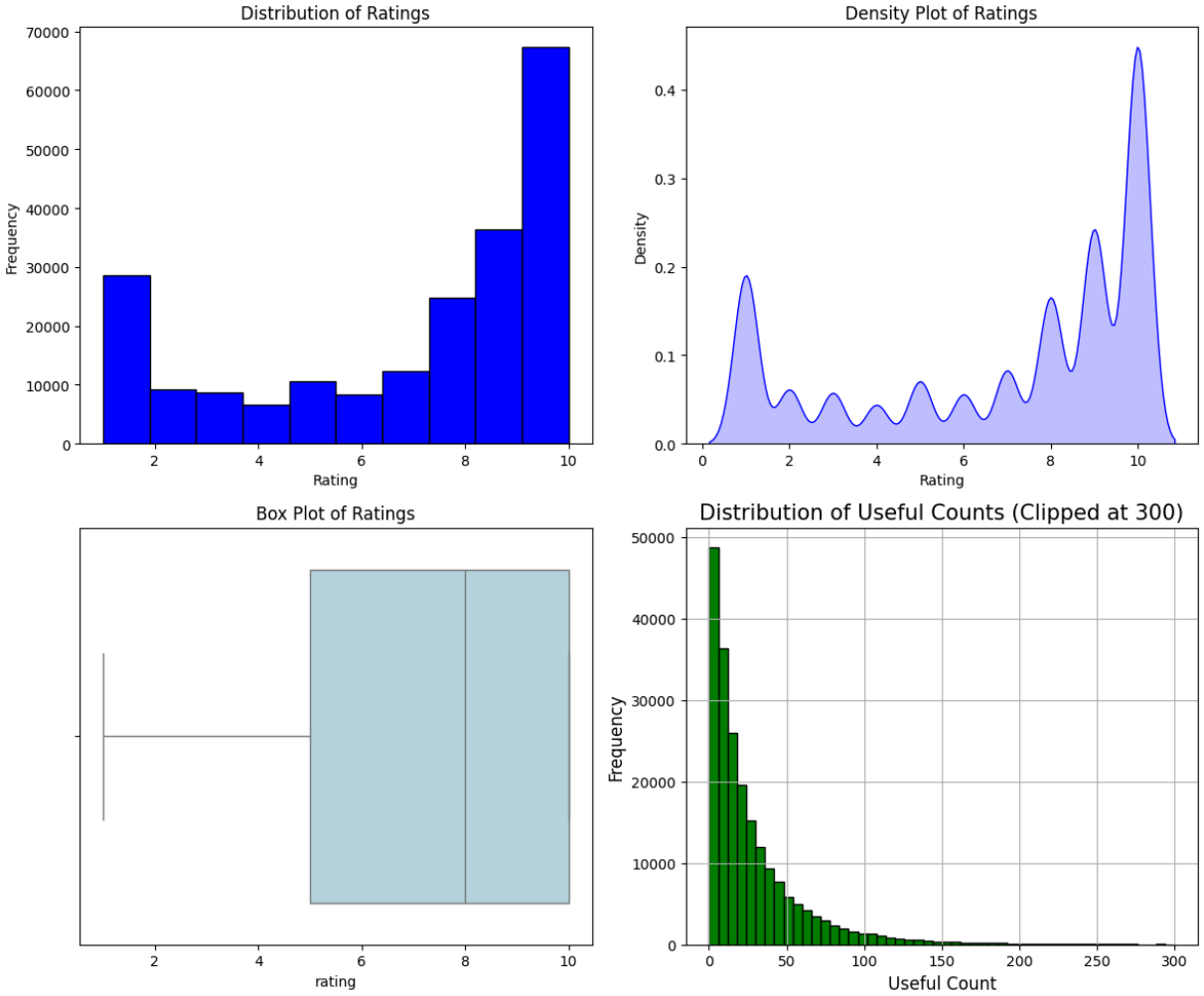
```
print(f"Number of unique drugs: {drug_count}")
print(f"Number of unique conditions: {condition_count}")
print(f"Average length of review: {round(average_review_length)}")
```

```
Missing Values
id                0
drugName          0
condition      1194
review            0
rating            0
date              0
usefulCount       0
dtype: int64

Number of rows with "</span>": 1171
Numerical Summary:
                  id         rating    usefulCount
count  212698.000000  212698.000000  212698.000000
mean   116079.388673       6.992431      28.186819
std     66999.171961       3.275994      36.455651
min         0.000000       1.000000       0.000000
25%     58154.250000       5.000000       6.000000
50%    115961.500000       8.000000      16.000000
75%    174009.750000      10.000000      37.000000
max    232291.000000      10.000000    1291.000000

Number of unique drugs: 3654
Number of unique conditions: 836
Average length of review: 85
```

In [5]:
```
# Convert 'date' column to datetime format
data_original['date'] = pd.to_datetime(data_original['date'], format="%d-%b-

# Extract year from the date
data_original['year'] = data_original['date'].dt.year
# Extract the month from the date
data_original['month'] = data_original['date'].dt.month

# Group data by year and month, and calculate the number of reviews and aver
data_original['year_month'] = data_original['date'].dt.to_period('M')
```

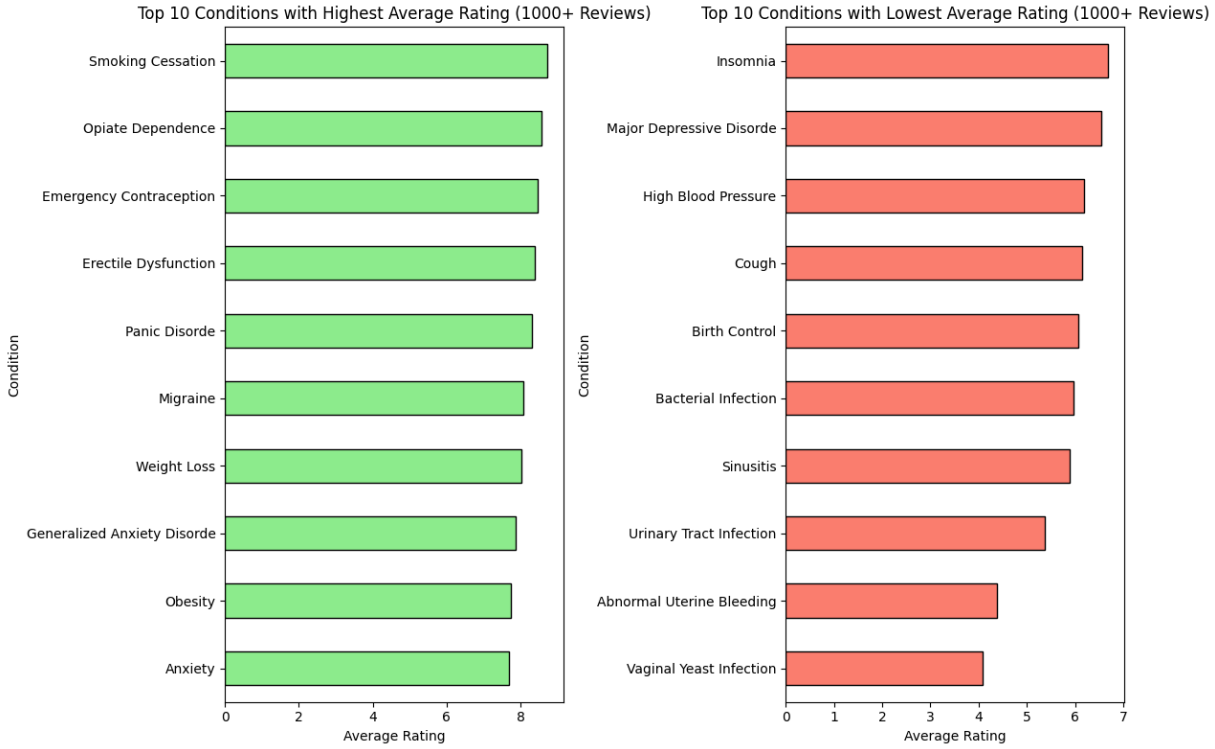
In [6]:
```
# Histogram for numerical variables (rating and usefulCount, assuming they e
fig, axs = plt.subplots(2, 2, figsize=(12, 10))

axs[0, 0].hist(data_original["rating"], bins=10, edgecolor="black", color="b
axs[0, 0].set_title('Distribution of Ratings')
axs[0, 0].set_xlabel('Rating')
axs[0, 0].set_ylabel('Frequency')

sns.kdeplot(data_original['rating'], fill=True, color='blue', ax=axs[0, 1])
axs[0, 1].set_title('Density Plot of Ratings')
axs[0, 1].set_xlabel('Rating')
axs[0, 1].set_ylabel('Density')
plt.grid(True)

# Box plots to show spread and outliers
```

```
sns.boxplot(x=data_original["rating"], color="lightblue", ax=axs[1, 0])
axs[1, 0].set_title("Box Plot of Ratings")

# Distribution of Useful counts plot
axs[1, 1].hist(data_original['usefulCount'], bins=50, color='green', edgecol
axs[1, 1].set_title('Distribution of Useful Counts (Clipped at 300)', fontsi
axs[1, 1].set_xlabel('Useful Count', fontsize=12)
axs[1, 1].set_ylabel('Frequency', fontsize=12)
axs[1, 1].grid(True)

plt.tight_layout()
plt.show()
```



In [7]:
```
# Calculate the average rating for each condition
avg_rating_per_condition = data_original.groupby("condition")["rating"].mean
condition_review_counts = data_original.groupby("condition").size()  # Count

conditions_at_least_1000_reviews = condition_review_counts[condition_review_
filtered_avg_ratings = avg_rating_per_condition.loc[conditions_at_least_1000

# Sort the conditions by average rating
sorted_ratings = filtered_avg_ratings.sort_values()

# Visualize the top 10 highest and lowest rated conditions with at least 100
plt.figure(figsize=(12, 8))
```

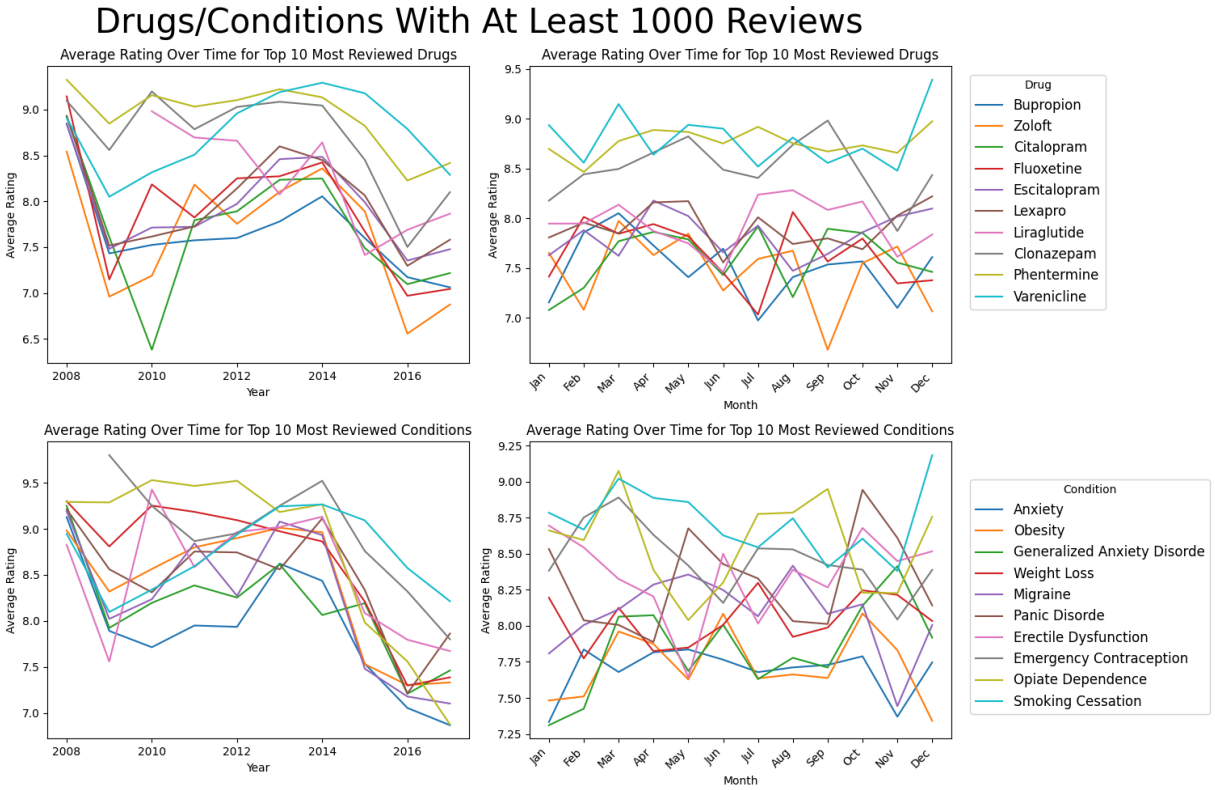
```
# Plot the top 10 highest rated conditions
plt.subplot(1, 2, 1)
sorted_ratings.tail(10).plot(kind="barh", color="lightgreen", edgecolor="bla
plt.title("Top 10 Conditions with Highest Average Rating (1000+ Reviews)")
plt.xlabel("Average Rating")
plt.ylabel("Condition")

# Plot the top 10 lowest rated conditions
plt.subplot(1, 2, 2)
sorted_ratings.head(10).plot(kind="barh", color="salmon", edgecolor="black")
plt.title("Top 10 Conditions with Lowest Average Rating (1000+ Reviews)")
plt.xlabel("Average Rating")
plt.ylabel("Condition")

# Display the plots
plt.tight_layout()
plt.show()
```



In [8]:
```
# string arr of all of the months in a year, used for the x ticks
months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oc
#Including the effect of time aka including the date information
avg_rating_per_drug = data_original.groupby("drugName")["rating"].mean()
drug_review_counts = data_original.groupby("drugName").size()

drugs_at_least_1000_reviews = drug_review_counts[drug_review_counts >= 1000]
filtered_avg_drug_ratings = avg_rating_per_drug.loc[drugs_at_least_1000_revi

# Sort the conditions by average rating
sorted_drug_ratings = filtered_avg_drug_ratings.sort_values()

# Identify the top 10 most used drugs (based on number of reviews)
top_10_drugs = sorted_drug_ratings.tail(10).index
```

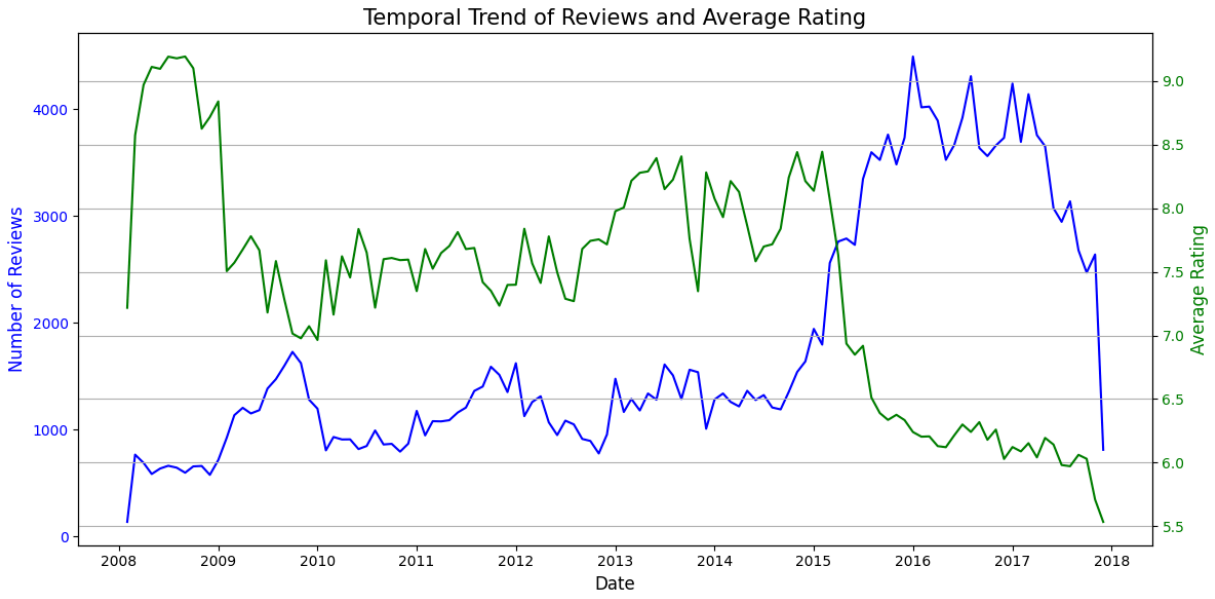
```
top_10_conditions = sorted_ratings.tail(10).index

# Group by drug and year/month, then calculate the average rating
ratings_by_year_top_drugs = data_original[data_original['drugName'].isin(top
ratings_by_month_top_drugs = data_original[data_original['drugName'].isin(to

# Group by condition and year/month, then calculate the average rating
ratings_by_year_top_conditions = data_original[data_original['condition'].is
ratings_by_month_top_conditions = data_original[data_original['condition'].i


# Create a 2x2 grid of subplots
fig, axs = plt.subplots(2, 2, figsize=(12, 10))
fig.suptitle('Drugs/Conditions With At Least 1000 Reviews', fontsize=30)

# Plotting for each drug and condition
for i in range(len(top_10_drugs)):
    drug = top_10_drugs[i]
    sns.lineplot(x='year', y='rating', data=ratings_by_year_top_drugs[rating
    sns.lineplot(x='month', y='rating', data=ratings_by_month_top_drugs[rati

    condition = top_10_conditions[i]
    sns.lineplot(x='year', y='rating', data=ratings_by_year_top_conditions[r
    sns.lineplot(x='month', y='rating', data=ratings_by_month_top_conditions

axs[0, 0].set_title('Average Rating Over Time for Top 10 Most Reviewed Drugs
axs[0, 0].set_xlabel('Year')
axs[0, 0].set_ylabel('Average Rating')

axs[0, 1].set_title('Average Rating Over Time for Top 10 Most Reviewed Drugs
axs[0, 1].set_xlabel('Month')
axs[0, 1].set_ylabel('Average Rating')
axs[0, 1].set_xticks(range(1, 13))  # Set ticks for each month (1 to 12)
axs[0, 1].set_xticklabels(months)

axs[1, 0].set_title('Average Rating Over Time for Top 10 Most Reviewed Condi
axs[1, 0].set_xlabel('Year')
axs[1, 0].set_ylabel('Average Rating')

axs[1, 1].set_title('Average Rating Over Time for Top 10 Most Reviewed Condi
axs[1, 1].set_xlabel('Month')
axs[1, 1].set_ylabel('Average Rating')
axs[1, 1].set_xticks(range(1, 13))  # Set ticks for each month (1 to 12)
axs[1, 1].set_xticklabels(months)

# Collect handles and labels from one of the subplots
handles_drugs, labels_drugs = axs[0, 0].get_legend_handles_labels()
handles_condition, labels_condition = axs[1, 0].get_legend_handles_labels()

# Remove the legends as it will be displayed on the side
axs[0, 0].legend().remove()
axs[0, 1].legend().remove()
axs[1, 0].legend().remove()
axs[1, 1].legend().remove()
```

```
# Set a common legend outside the plot (to the right)
fig.legend(handles_drugs, labels_drugs, loc='center left', bbox_to_anchor=(1
fig.legend(handles_condition, labels_condition, loc='center left', bbox_to_a

# Rotate the x-ticks to make them readable
plt.setp(axs[0, 1].get_xticklabels(), rotation=45, ha="right")
plt.setp(axs[1, 1].get_xticklabels(), rotation=45, ha="right")

plt.tight_layout()
plt.show()
```

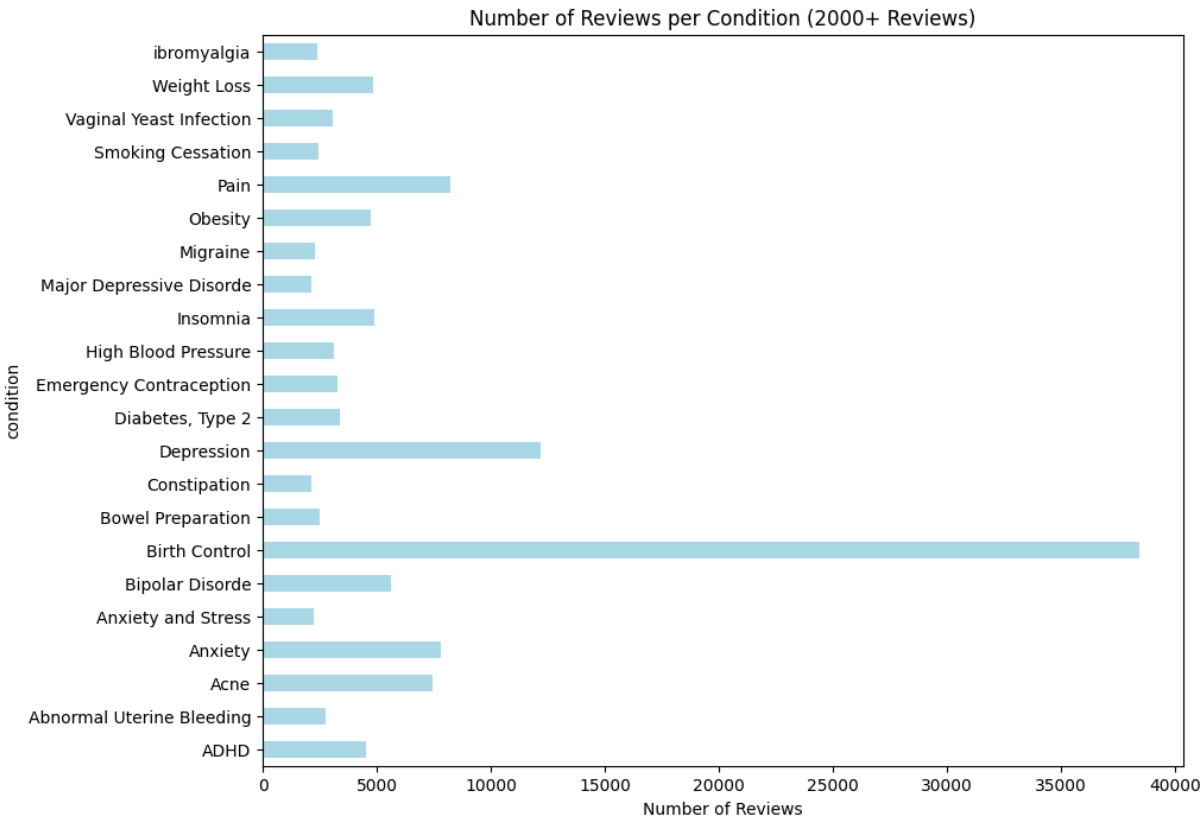


In [9]:
```
# Aggregating data by month (number of reviews and average rating)
temporal_data = data_original.groupby('year_month').agg(
    num_reviews=('review', 'count'),
    avg_rating=('rating', 'mean')
).reset_index()

# Convert year_month back to datetime for proper plotting
temporal_data['year_month'] = temporal_data['year_month'].dt.to_timestamp()

# Plotting temporal trends
fig, ax1 = plt.subplots(figsize=(12, 6))

# Plot number of reviews over time
ax1.plot(temporal_data['year_month'], temporal_data['num_reviews'], color='b
ax1.set_xlabel('Date', fontsize=12)
ax1.set_ylabel('Number of Reviews', fontsize=12, color='blue')
ax1.tick_params(axis='y', labelcolor='blue')

# Create a second y-axis for the average rating
ax2 = ax1.twinx()
ax2.plot(temporal_data['year_month'], temporal_data['avg_rating'], color='gr
```

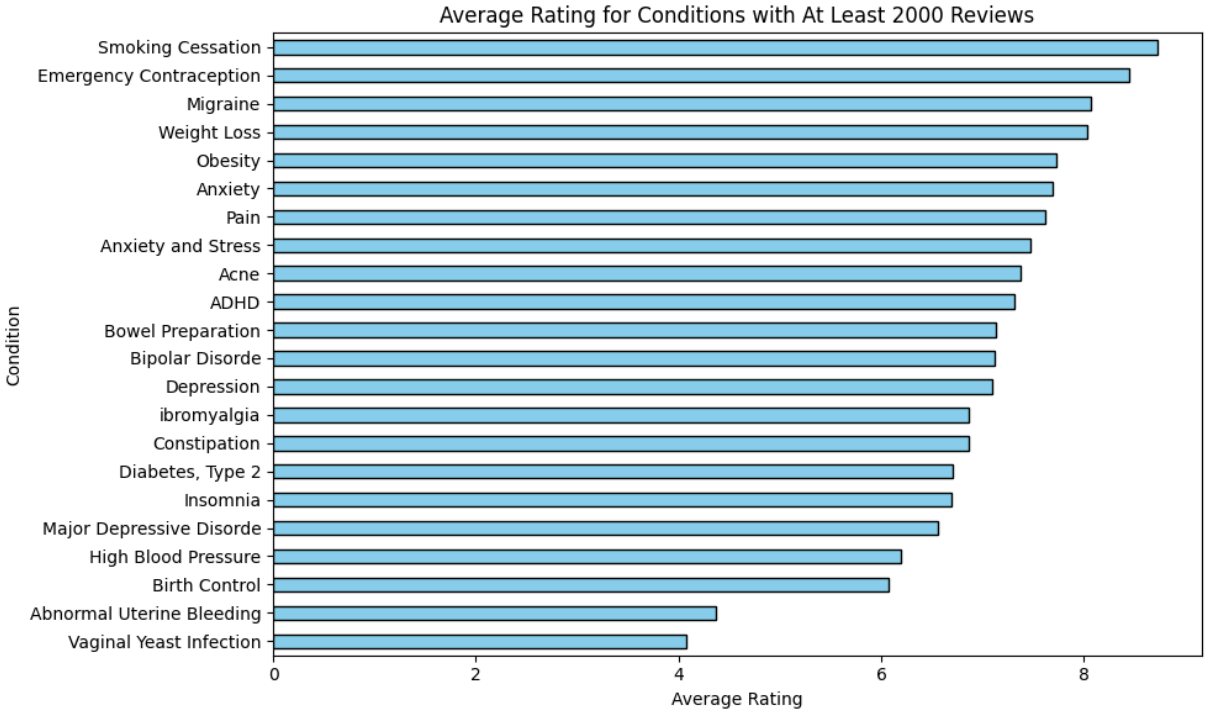
```
ax2.set_ylabel('Average Rating', fontsize=12, color='green')
ax2.tick_params(axis='y', labelcolor='green')

plt.title('Temporal Trend of Reviews and Average Rating', fontsize=15)
plt.grid(True)
plt.tight_layout()

plt.show()
```



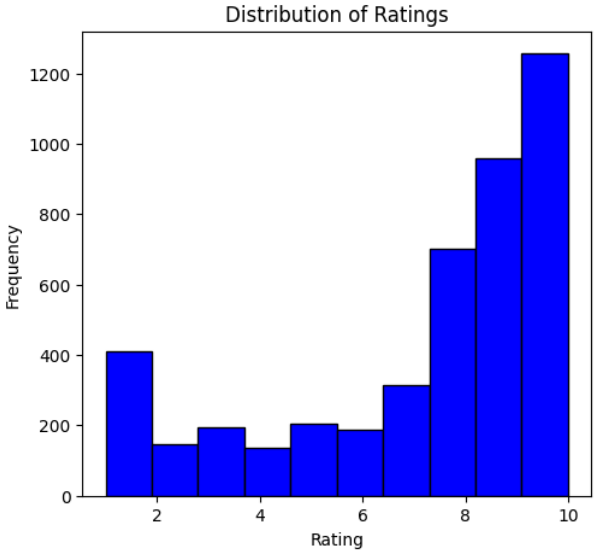
In [10]:
```
conditions_large_reviews = condition_review_counts[condition_review_counts >

plt.figure(figsize=(12, 5))
conditions_large_reviews.plot(kind="barh", figsize=(10, 8), color="lightblue
plt.title("Number of Reviews per Condition (2000+ Reviews)")
plt.xlabel("Number of Reviews")
plt.show()
```

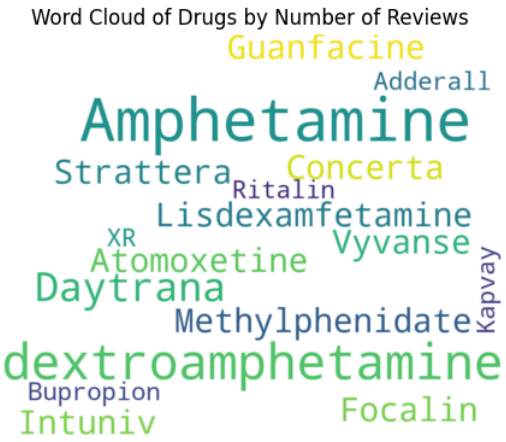


In [11]:
```
# Filter conditions with at least 2000 reviews
conditions_at_least_2000_reviews = condition_review_counts[condition_review_
filtered_avg_ratings_2000 = avg_rating_per_condition.loc[conditions_at_least

plt.figure(figsize=(10, 6))
filtered_avg_ratings_2000.sort_values().plot(kind="barh", color="skyblue", e
plt.title("Average Rating for Conditions with At Least 2000 Reviews")
plt.xlabel("Average Rating")
plt.ylabel("Condition")
plt.tight_layout()

# Show the plot
plt.show()
```



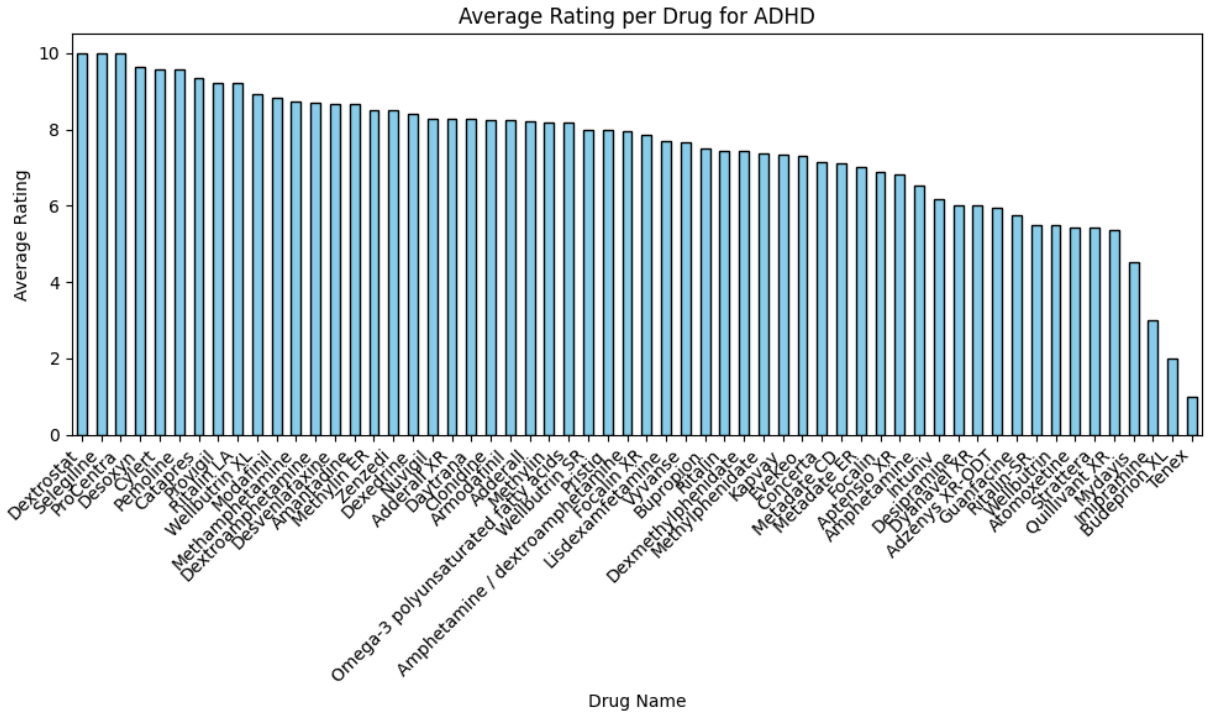
In [12]:
```
adhd = data_original[data_original["condition"] == "ADHD"]

plt.figure(figsize=(12, 5))

# Histogram for "rating"
plt.subplot(1, 2, 1)
plt.hist(adhd["rating"], bins=10, color="blue", edgecolor="black")
plt.title("Distribution of Ratings")
plt.xlabel("Rating")
plt.ylabel("Frequency")


# Generate a string of all drug names repeated based on their counts
drug_counts = adhd["drugName"].head(30).value_counts()
drug_string = " ".join([f"{drug} " * count for drug, count in drug_counts.it

# Create the word cloud
wordcloud = WordCloud(width=1000, height=800, background_color="white").gene

# Plot the word cloud
plt.subplot(1, 2, 2)
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")  # Turn off the axis
plt.title("Word Cloud of Drugs by Number of Reviews")
plt.show()

# Display the plots
plt.tight_layout()
plt.show()
```



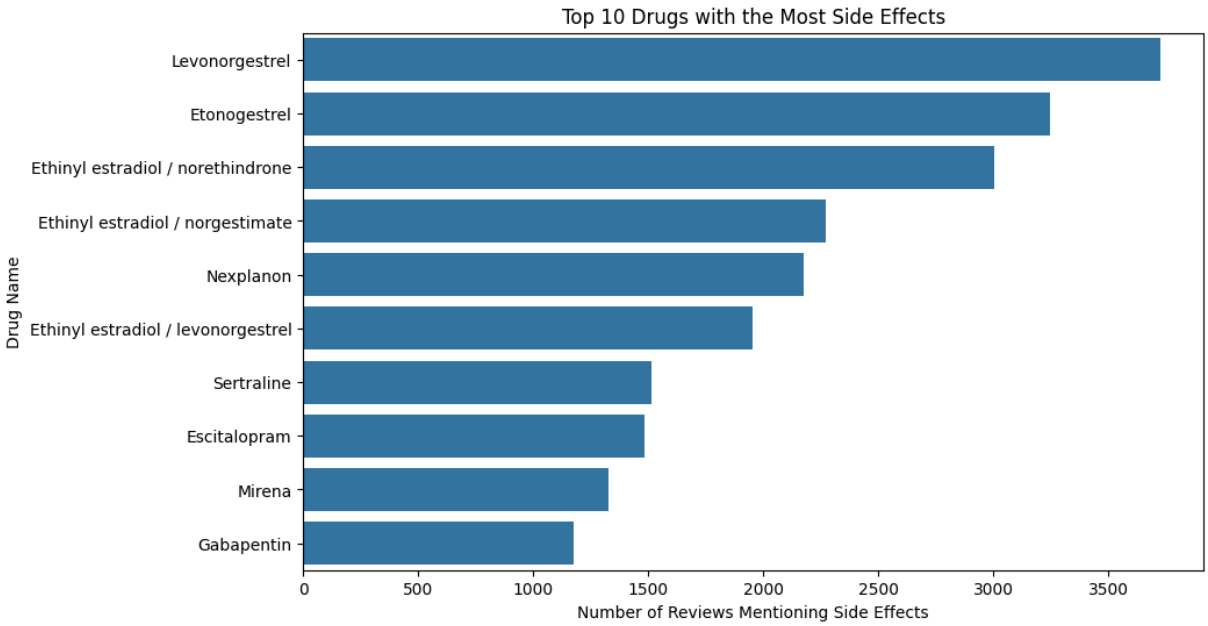


<Figure size 640x480 with 0 Axes>

In [13]:
```
# Calculate the average rating for each drug
avg_rating_per_drug = adhd.groupby("drugName")["rating"].mean()

# Plotting the average rating per drug
plt.figure(figsize=(10, 6))
avg_rating_per_drug.sort_values(ascending=False).plot(kind="bar", color="sky
plt.title("Average Rating per Drug for ADHD")
plt.xlabel("Drug Name")
plt.ylabel("Average Rating")
plt.xticks(rotation=45, ha="right")
plt.tight_layout()

# Show the plot
plt.show()
```



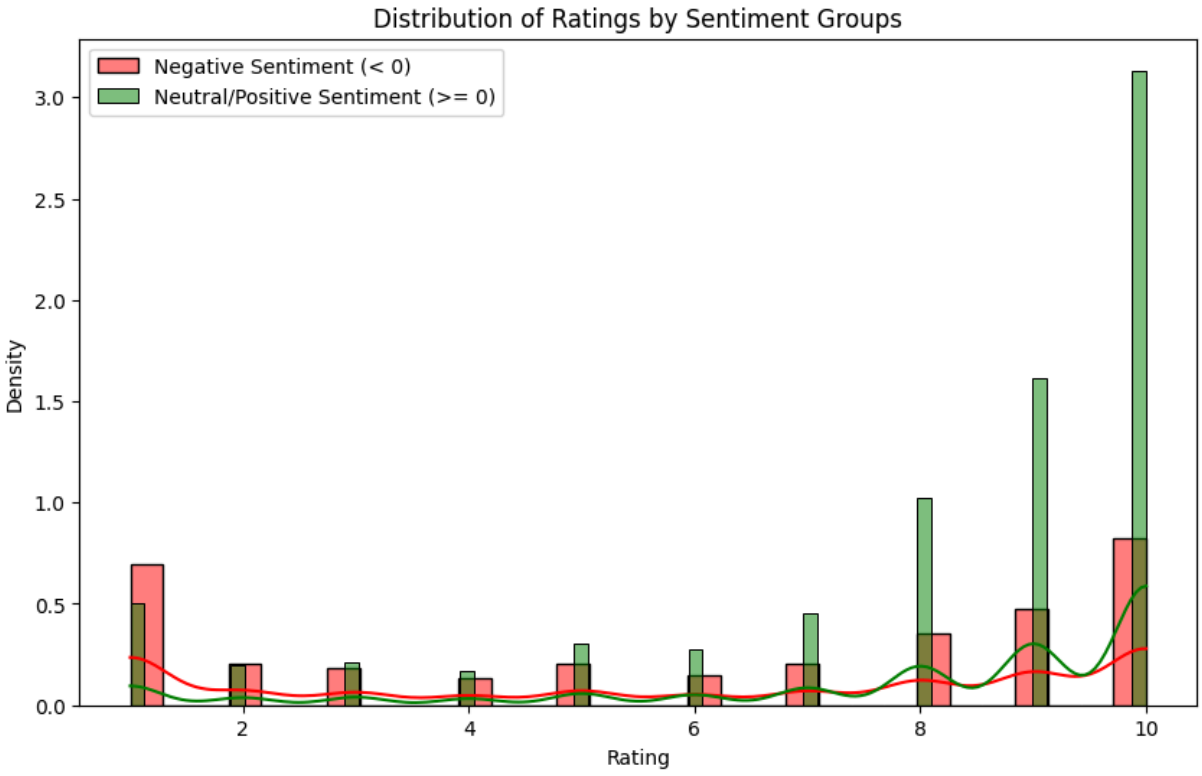
In [14]:
```
# Import local module helper function that checks if side effects are mentic
# See Appendix B
from utils.side_effects import check_side_effects

# Apply the function to the review text and create the 'side_effects' columr
data_original['side_effects'] = data_original['review'].apply(check_side_eff

# Check the distribution of the new target variable
side_effect_counts = data_original['side_effects'].value_counts()
#print(f"Side Effect Target Distribution:\n{side_effect_counts}")

# Group by drug name and sum the 'side_effects' column to count how many rev
side_effects_by_drug = data_original.groupby('drugName')['side_effects'].sum

# Sort the results to find the drugs with the most side effects
side_effects_by_drug_sorted = side_effects_by_drug.sort_values(by='side_effe

# Plot the top 10 drugs with the most side effects
plt.figure(figsize=(10, 6))
sns.barplot(x='side_effects', y='drugName', data=side_effects_by_drug_sortec
plt.title('Top 10 Drugs with the Most Side Effects')
plt.xlabel('Number of Reviews Mentioning Side Effects')
plt.ylabel('Drug Name')
plt.show()
```



In [15]:
```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from sklearn.model_selection import train_test_split
sid = SentimentIntensityAnalyzer()

# Split the data into 75% training and 25% testing
train_df, test_df = train_test_split(data_original, test_size=0.25, random_s
# Preprocess text for sentiment analysis (get sentiment score from review te
data_original['sentiment'] = data_original['review'].apply(lambda x: sid.pol

# Separate the data into two groups: negative (sentiment < 0) and neutral/po
negative = data_original[data_original['sentiment'] < 0]
```

```
neutral_positive = data_original[data_original['sentiment'] >= 0]

avg_rating_negative = negative['rating'].mean()
avg_rating_neutral_positive = neutral_positive['rating'].mean()

# Print the results
print(f"Average rating for negative sentiment (sentiment < 0): {avg_rating_r
print(f"Average rating for neutral/positive sentiment (sentiment >= 0): {avg

# plot the distribution of ratings for each sentiment group
plt.figure(figsize=(10, 6))
sns.histplot(negative['rating'], color='red', label='Negative Sentiment (< 0
sns.histplot(neutral_positive['rating'], color='green', label='Neutral/Posit
plt.title('Distribution of Ratings by Sentiment Groups')
plt.xlabel('Rating')
plt.ylabel('Density')
plt.legend()
plt.show()
```

```
Average rating for negative sentiment (sentiment < 0): 6.08
Average rating for neutral/positive sentiment (sentiment >= 0): 7.94
```



In [16]:

```
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# Define effectiveness for training data, here we assumed the ratings > 5 me
# which can be adjusted withe the previous section of sentiment + rating
train_df['effective'] = train_df.apply(
    lambda row: 1 if row['rating'] > 5 else 0, axis=1
)

# Define effectiveness for test data in the same way
```

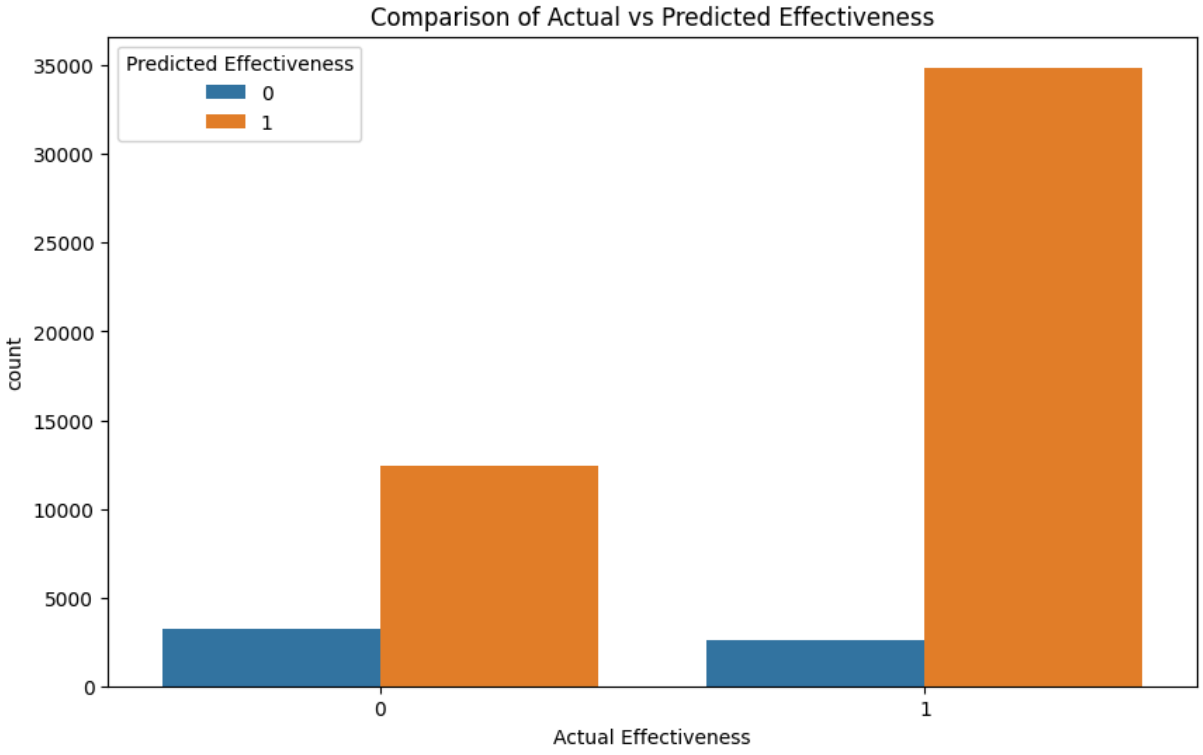
```
test_df['effective'] = test_df.apply(
    lambda row: 1 if row['rating'] > 5 else 0, axis=1
)

# Handle missing data
train_df.dropna(subset=['drugName', 'condition', 'rating'], inplace=True)
test_df.dropna(subset=['drugName', 'condition', 'rating'], inplace=True)

# Encode categorical variables (Drug name, Condition)
le_drug = LabelEncoder()
le_condition = LabelEncoder()

train_df['drug_encoded'] = le_drug.fit_transform(train_df['drugName'])
train_df['condition_encoded'] = le_condition.fit_transform(train_df['conditi

# Define a safe transformation function to handle unseen labels
def safe_transform(encoder, data, default_value=-1):
    return np.array([default_value if label not in encoder.classes_ else enc

# Apply safe transformation on the test data to handle unseen drugs and cond
test_df['drug_encoded'] = safe_transform(le_drug, test_df['drugName'])
test_df['condition_encoded'] = safe_transform(le_condition, test_df['conditi

# Define the input features (drug name and condition) and the target variabl
X_train = train_df[['drug_encoded', 'condition_encoded']]
y_train = train_df['effective']

X_test = test_df[['drug_encoded', 'condition_encoded']]
y_test = test_df['effective']

# Train a Random Forest classifier using drug name and condition as input fe
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)

# Make predictions on the test set using drug name and condition
y_pred = clf.predict(X_test)

# Evaluate the model
print(f"Train Accuracy: {accuracy_score(y_train, clf.predict(X_train))}")
print(f"Test Accuracy: {accuracy_score(y_test, y_pred)}")
print(classification_report(y_test, y_pred))

# Create a comparison table of actual vs predicted values
comparison_df = pd.DataFrame({
    'Drug Name': test_df['drugName'],
    'Condition': test_df['condition'],
    'Actual Effectiveness': y_test,
    'Predicted Effectiveness': y_pred
})

print(comparison_df.head())

# Plot the predicted effectiveness for a few drug-condition pairs
plt.figure(figsize=(10, 6))
sns.countplot(x='Actual Effectiveness', hue='Predicted Effectiveness', data=
```

```
plt.title('Comparison of Actual vs Predicted Effectiveness')
plt.show()
```

```
Train Accuracy: 0.7290484757683845
Test Accuracy: 0.7173859896567936
              precision    recall  f1-score   support

           0       0.56      0.21      0.30     15697
           1       0.74      0.93      0.82     37478

    accuracy                           0.72     53175
   macro avg       0.65      0.57      0.56     53175
weighted avg       0.68      0.72      0.67     53175
```

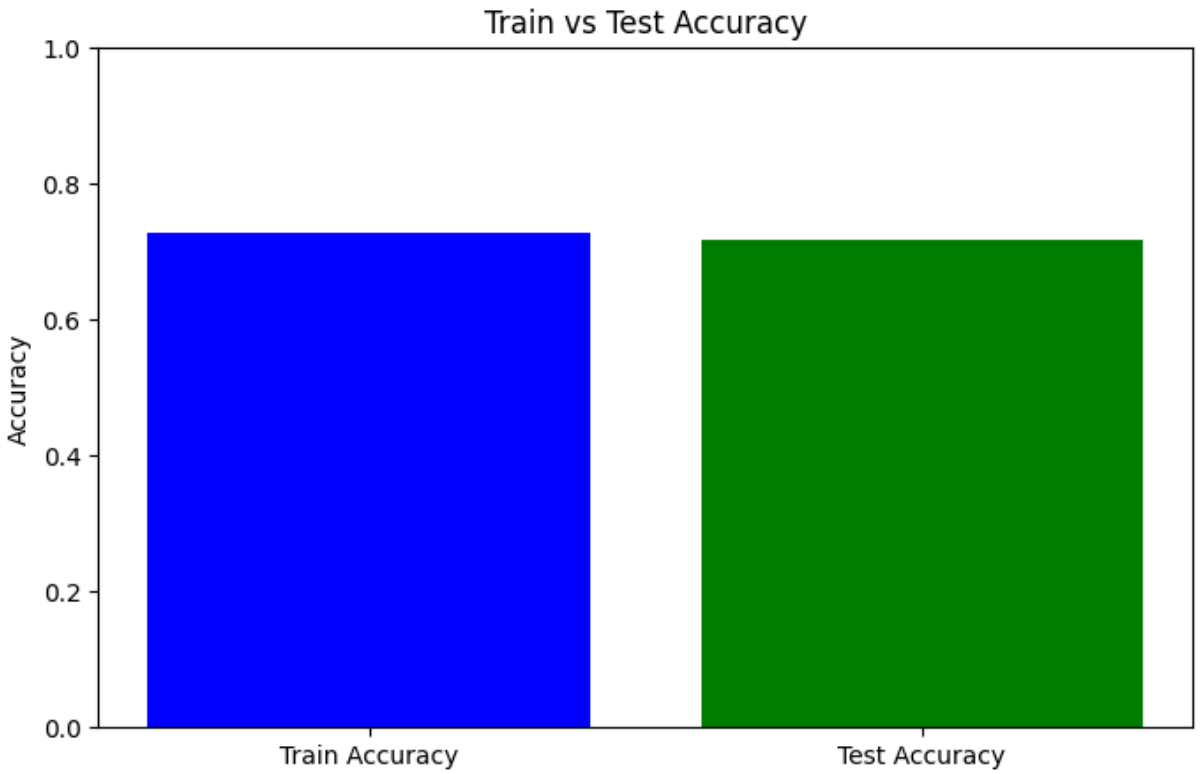
	Drug Name	Condition	\
212040	Ethinyl estradiol / etonogestrel	Birth Control	
177292	Dextromethorphan	Cough	
117623	Topamax	Migraine Prevention	
32740	Phentermine / topiramate	Weight Loss	
193096	Vasotec	Left Ventricular Dysfunction	

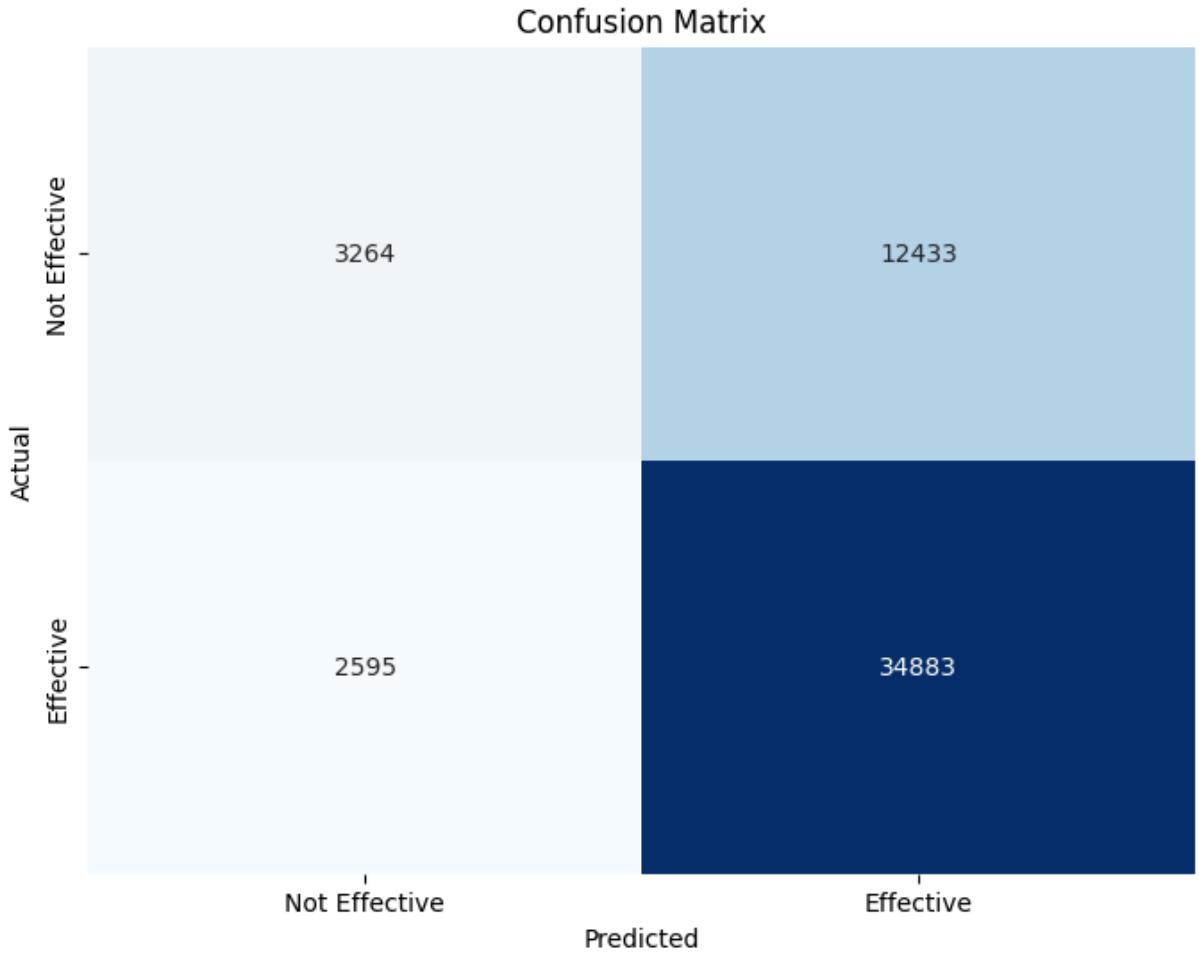
	Actual Effectiveness	Predicted Effectiveness
212040	0	1
177292	0	0
117623	0	1
32740	1	1
193096	1	1

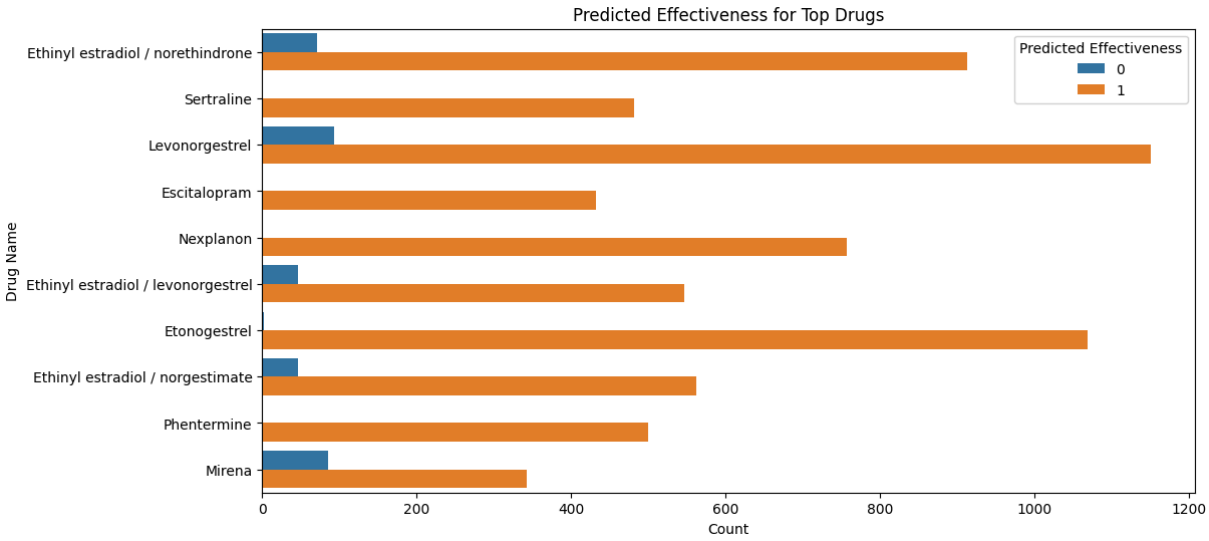


In [17]:
```
#Some visualizations, we can pick and chose what we want to keep
from sklearn.metrics import confusion_matrix
import seaborn as sns

# Train and Test Accuracy as a bar plot
plt.figure(figsize=(8, 5))
```

```
accuracy_values = [accuracy_score(y_train, clf.predict(X_train)), accuracy_s
plt.bar(['Train Accuracy', 'Test Accuracy'], accuracy_values, color=['blue',
plt.title('Train vs Test Accuracy')
plt.ylabel('Accuracy')
plt.ylim(0, 1)
plt.show()

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False, xticklabels=[
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

# Bar plot of Predicted Effectiveness for different drugs
plt.figure(figsize=(12, 6))
top_drugs = comparison_df['Drug Name'].value_counts().index[:10]
sns.countplot(y='Drug Name', hue='Predicted Effectiveness', data=comparison_
plt.title('Predicted Effectiveness for Top Drugs')
plt.xlabel('Count')
plt.ylabel('Drug Name')
plt.legend(title='Predicted Effectiveness', loc='upper right')
plt.show()
```



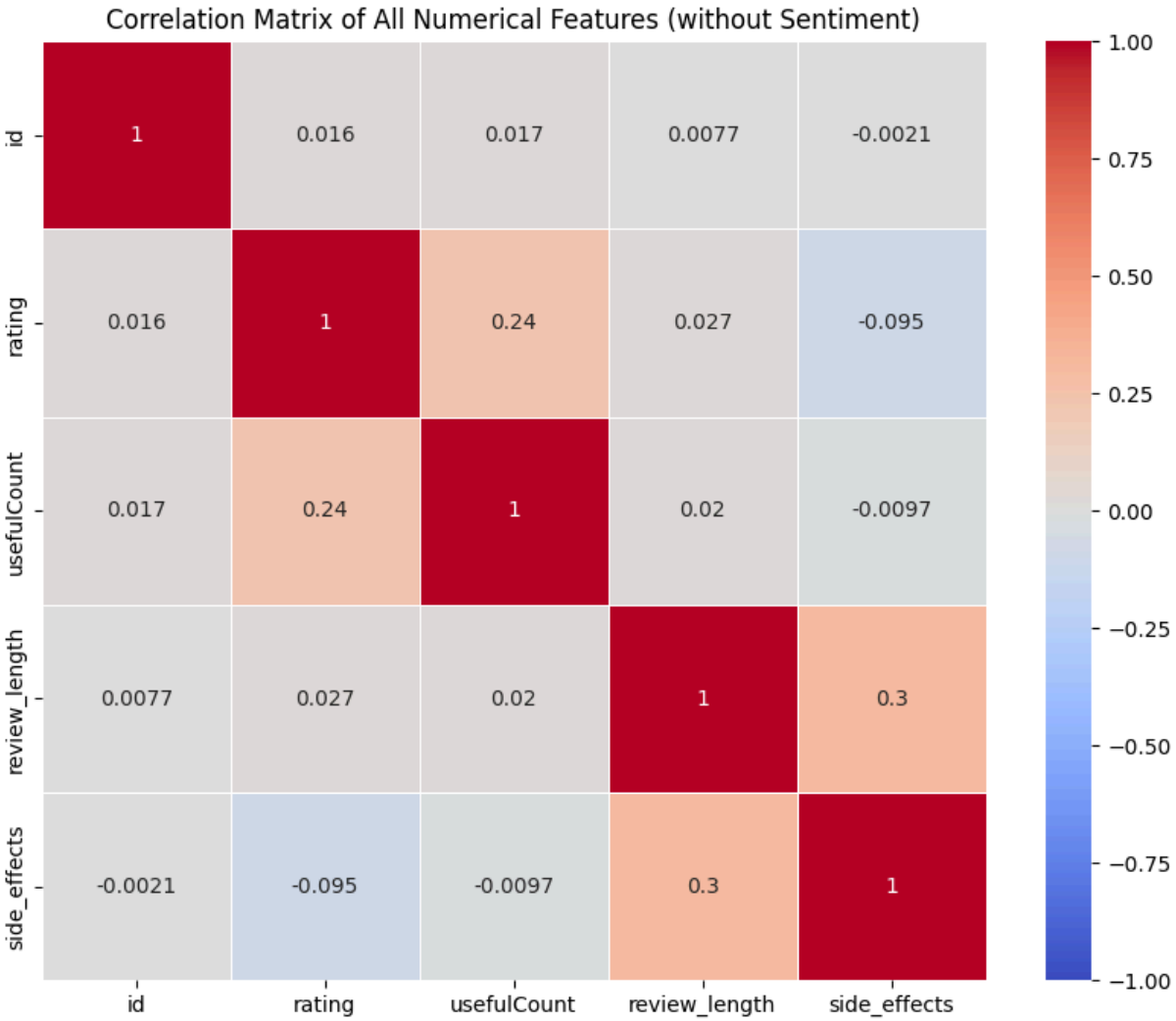


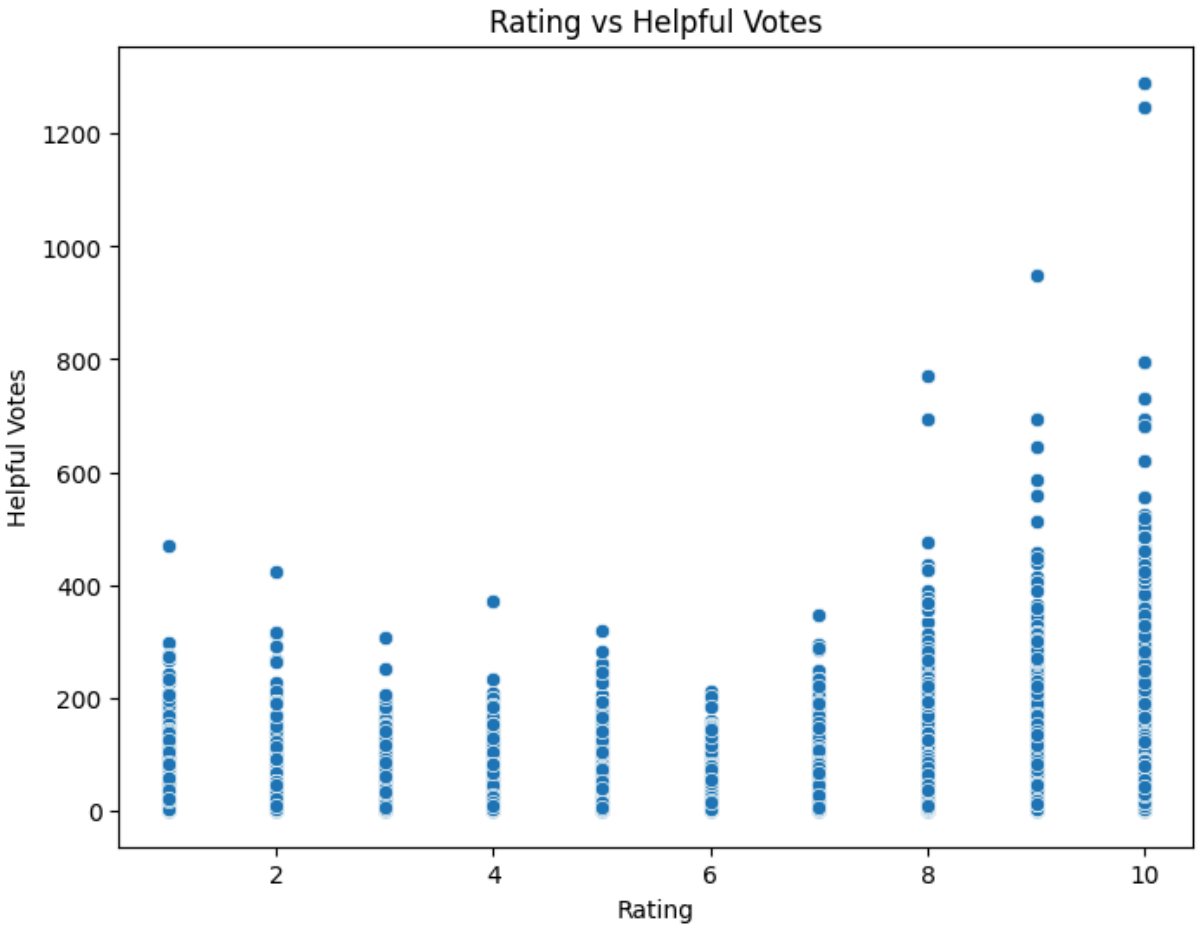


In [18]:
```python
# Some additional descriptive analysis that could be useful:
# Select all numerical columns except 'sentiment'
numerical_columns = data_original.select_dtypes(include=['float64', 'int64']
numerical_columns = numerical_columns.drop('sentiment', errors='ignore')  #

# Calculate the correlation matrix for the selected numerical features
correlation_matrix = data_original[numerical_columns].corr()

# Display the correlation matrix
print("Correlation Matrix (without Sentiment):")
```

```
print(correlation_matrix)

# Plot a heatmap of the correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1
plt.title('Correlation Matrix of All Numerical Features (without Sentiment)'
plt.show()

# Scatter plot for rating vs usefulCount (helpful votes)
plt.figure(figsize=(8, 6))
sns.scatterplot(x='rating', y='usefulCount', data=data_original)
plt.title('Rating vs Helpful Votes')
plt.xlabel('Rating')
plt.ylabel('Helpful Votes')
plt.show()
```

```
Correlation Matrix (without Sentiment):
                     id    rating  usefulCount  review_length  side_effects
id             1.000000  0.016090     0.017253       0.007651     -0.002111
rating         0.016090  1.000000     0.235375       0.027199     -0.094916
usefulCount    0.017253  0.235375     1.000000       0.019819     -0.009713
review_length  0.007651  0.027199     0.019819       1.000000      0.303048
side_effects  -0.002111 -0.094916    -0.009713       0.303048      1.000000
```

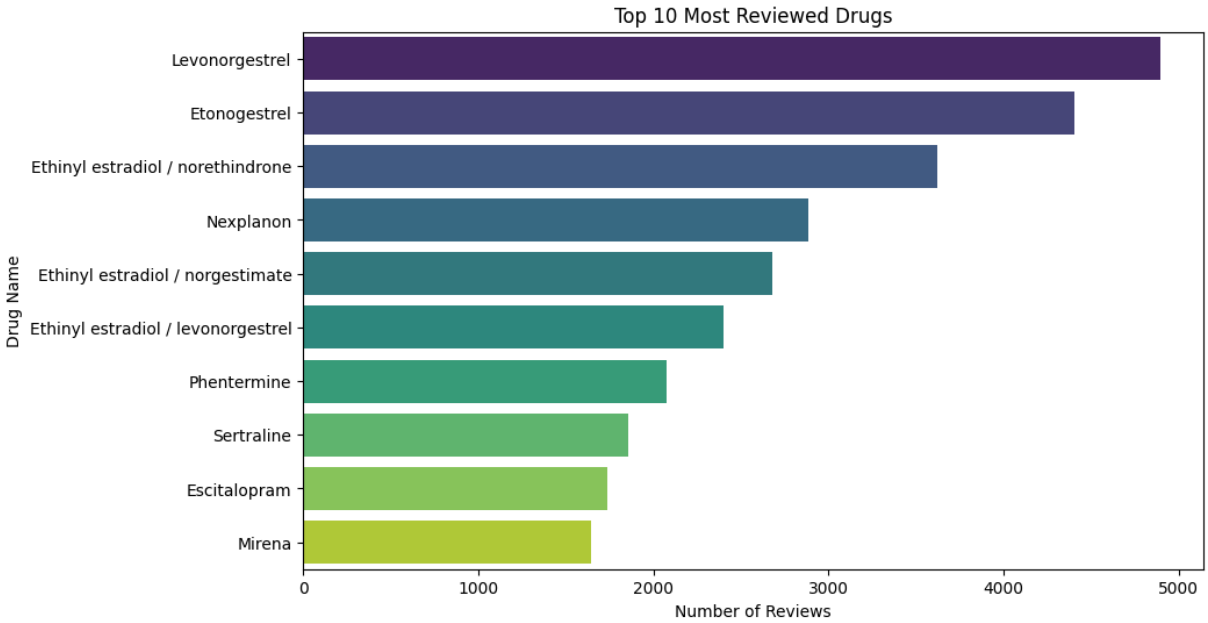




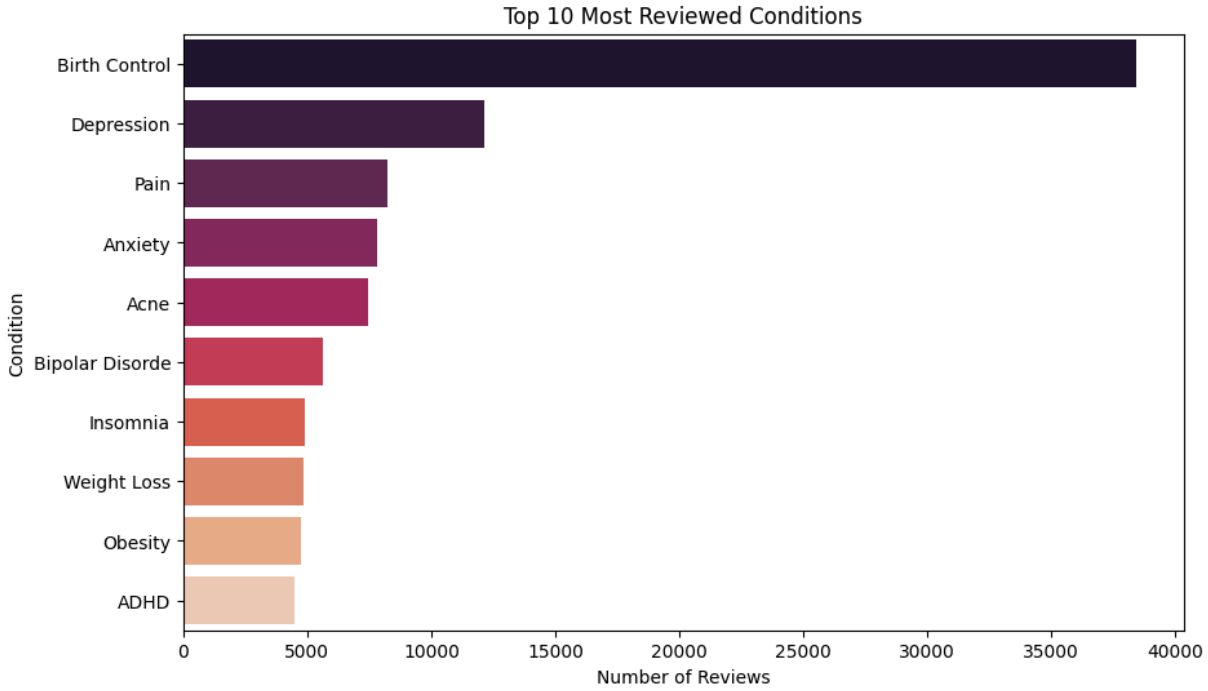
In [19]:
```
# Top 10 most common drugs
top_drugs = data_original['drugName'].value_counts().head(10)
print("Top 10 Most Reviewed Drugs:")
print(top_drugs)

# Visualize the top 10 most common drugs
plt.figure(figsize=(10, 6))
sns.barplot(x=top_drugs.values, y=top_drugs.index, hue=top_drugs.index, lege
plt.title('Top 10 Most Reviewed Drugs')
plt.xlabel('Number of Reviews')
plt.ylabel('Drug Name')
plt.show()

# Top 10 most common conditions
top_conditions = data_original['condition'].value_counts().head(10)
print("Top 10 Most Reviewed Conditions:")
print(top_conditions)

# Visualize the top 10 conditions
plt.figure(figsize=(10, 6))
sns.barplot(x=top_conditions.values, y=top_conditions.index, hue=top_conditi
plt.title('Top 10 Most Reviewed Conditions')
plt.xlabel('Number of Reviews')
plt.ylabel('Condition')
plt.show()

# Number of reviews per drug and condition
```

```python
reviews_per_drug_condition = data_original.groupby(['drugName', 'condition']
top_drug_condition = reviews_per_drug_condition.sort_values(by='num_reviews'

print("Top 10 Drug-Condition Pairs by Number of Reviews:")
print(top_drug_condition)

# Visualize the top drug-condition pairs
plt.figure(figsize=(10, 6))
sns.barplot(x='num_reviews', y='drugName', hue='condition', data=top_drug_cc
plt.title('Top 10 Drug-Condition Pairs by Number of Reviews')
plt.xlabel('Number of Reviews')
plt.ylabel('Drug Name')
plt.legend(title='Condition', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()

# Plot the distribution of sentiment scores
plt.figure(figsize=(10, 6))
sns.histplot(data_original['sentiment'], bins=20, kde=True, color='purple')
plt.title('Sentiment Score Distribution')
plt.xlabel('Sentiment Score')
plt.ylabel('Count')
plt.show()

# Show how many reviews have positive, negative, or neutral sentiment
print("Sentiment Distribution:")
sentiment_labels = ['Negative', 'Neutral', 'Positive']
sentiment_distribution = [
    (data_original['sentiment'] < 0).sum(),
    (data_original['sentiment'] == 0).sum(),
    (data_original['sentiment'] > 0).sum()
]
for label, count in zip(sentiment_labels, sentiment_distribution):
    print(f"{label}: {count} reviews")
```

```
Top 10 Most Reviewed Drugs:
drugName
Levonorgestrel                        4896
Etonogestrel                          4402
Ethinyl estradiol / norethindrone     3619
Nexplanon                             2883
Ethinyl estradiol / norgestimate      2682
Ethinyl estradiol / levonorgestrel    2400
Phentermine                           2077
Sertraline                            1859
Escitalopram                          1739
Mirena                                1647
Name: count, dtype: int64
```

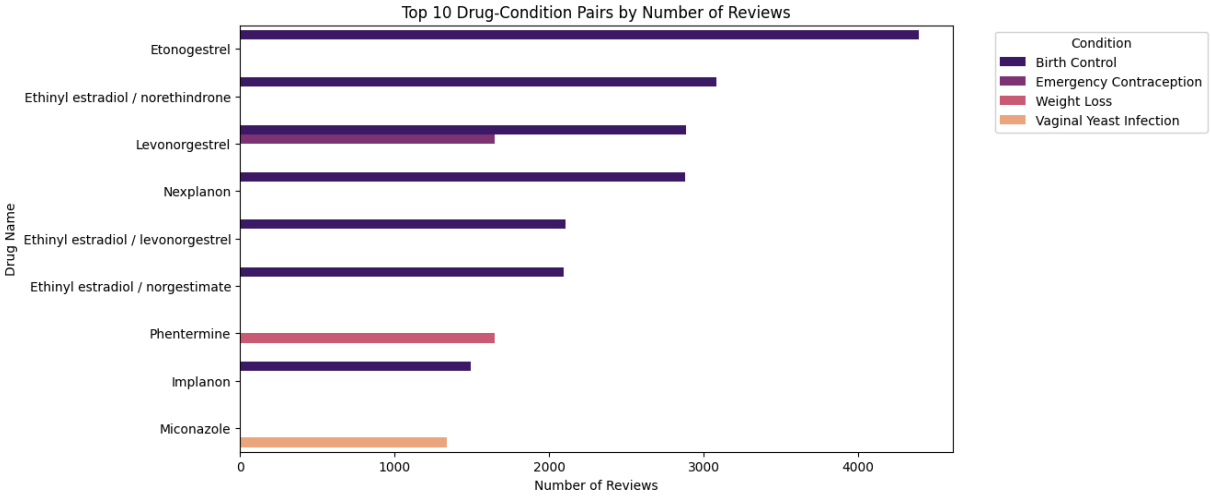


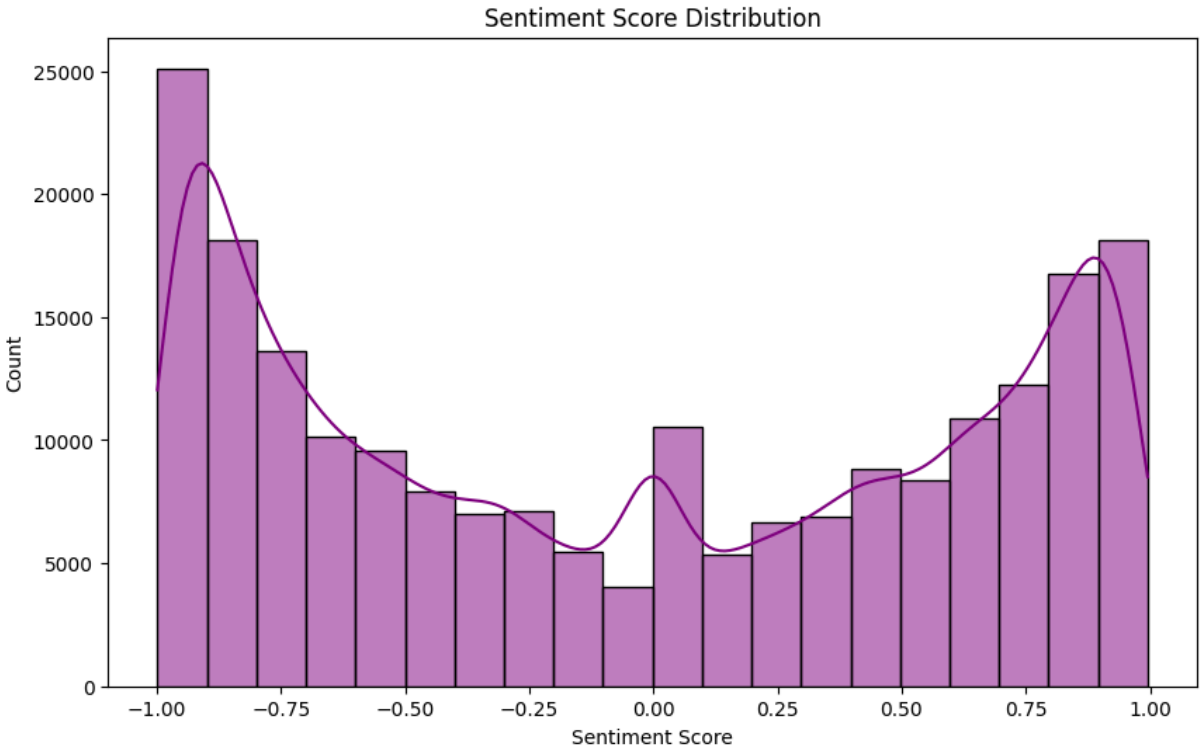

```
Top 10 Most Reviewed Conditions:
condition
Birth Control      38436
Depression         12164
Pain                8245
Anxiety             7812
Acne                7435
Bipolar Disorde     5604
Insomnia            4904
Weight Loss         4857
Obesity             4757
ADHD                4509
Name: count, dtype: int64
```



Top 10 Drug-Condition Pairs by Number of Reviews:

	drugName	condition	num_reviews
3073	Etonogestrel	Birth Control	4394
3049	Ethinyl estradiol / norethindrone	Birth Control	3081
4442	Levonorgestrel	Birth Control	2884
5520	Nexplanon	Birth Control	2883
3040	Ethinyl estradiol / levonorgestrel	Birth Control	2107
3058	Ethinyl estradiol / norgestimate	Birth Control	2097
4443	Levonorgestrel	Emergency Contraception	1651
6136	Phentermine	Weight Loss	1650
3943	Implanon	Birth Control	1496
5099	Miconazole	Vaginal Yeast Infection	1338






```
Sentiment Distribution:
Negative: 108118 reviews
Neutral: 6691 reviews
Positive: 97889 reviews
```

In [20]:

```python
import ipywidgets as widgets
from IPython.display import display

# Define effectiveness using percentiles of the ratings to add more variance
# Compute percentiles and scale effectiveness
data_original['percentile_effectiveness'] = data_original.groupby('condition

# Group by condition and drug, then calculate the mean percentile effectiven
effectiveness_by_condition = data_original.groupby(['condition', 'drugName']
    average_rating=('rating', 'mean'),
    confidence=('percentile_effectiveness', 'mean')  # Now using percentile
).reset_index()

# Function to get the best drug for a given condition
def get_best_drug(condition):
    condition_data = effectiveness_by_condition[effectiveness_by_condition['

    if condition_data.empty:
        return "No data available", 0

    best_drug = condition_data.sort_values(by='confidence', ascending=False)

    return best_drug['drugName'], best_drug['confidence']  # Return the conf

# Create a dropdown widget for selecting a condition
condition_dropdown = widgets.Dropdown(
    options=data_original['condition'].unique(),
    description='Condition:',
```

```
    disabled=False
)

# Create a label widget to display the best drug and confidence
result_label = widgets.Label(value="Select a condition to see the best drug"

# Function to update the result when a condition is selected
def on_condition_change(change):
    if change['type'] == 'change' and change['name'] == 'value':
        selected_condition = change['new']
        best_drug, confidence = get_best_drug(selected_condition)
        result_label.value = f"Best Drug: {best_drug}\nConfidence: {confider

# Attach the function to the dropdown widget
condition_dropdown.observe(on_condition_change)

# Display the dropdown and result label in the notebook
display(condition_dropdown)
display(result_label)
```

```
Dropdown(description='Condition:', options=('Left Ventricular Dysfunction',
'ADHD', 'Birth Control', 'Opiate D…
Label(value='Select a condition to see the best drug')
```

Appendix B: Check For Side Effects

```
import re
import sys
sys.path.append("./utils")

# import local constants and helpers. See Appendix C for Keywords.
from keywords import side_effect_keywords, negation_words
# See Appendix D for parser.
from parser import parse_string, find_first_occurence

def check_for_negation(text, keyword):
   words = text.split()
   keyword_index = find_first_occurence(keyword, words)

   if keyword_index == -1:
       return False

   # Search for negation within the 3 words before the keyword
   window = words[max(0, keyword_index - 3): keyword_index]

   # iterate through the negation words and check if any of them are found in the
window.
   for negation in negation_words:
       if any(negation in word for word in window):
           return True
   return False

# Create a function to check if the review mentions side effects or no side effects
def check_side_effects(review):
   if review == '':
       return 0

   review = parse_string(review) # Convert the review to lowercase

   # Search for any of the keywords in the review
   for keyword in side_effect_keywords:
       # Search for the keyword as a whole word using regular expressions
       if re.search(re.escape(keyword), review):
           # If the keyword is found, check for negation
           if not check_for_negation(review, keyword):
               return 1  # Side effect found and not negated
   return 0  # No side effects mentioned or negated
```

Appendix C: Keywords

```
side_effect_keywords = [
   # General symptoms
   'headache', 'migraine', 'dizzy', 'dizziness', 'lightheaded', 'faint', 'nausea',
'nauseous',
   'vomit', 'vomiting', 'upset stomach', 'stomach ache', 'abdominal pain', 'cramps',
'bloating',
   'diarrhea', 'constipation', 'indigestion', 'heartburn', 'gas', 'belching', 'burping',

   # Fatigue and energy
   'fatigue', 'tired', 'exhausted', 'sleepy', 'drowsy', 'lethargic', 'low energy',
'sluggish',
   'weakness', 'muscle weakness', 'muscle pain', 'joint pain', 'achy', 'aches', 'pain',
'back pain',

   # Mood and mental health
   'anxiety', 'anxious', 'panic attacks', 'depression', 'mood swings', 'irritable',
'agitation',
   'nervous', 'restless', 'confusion', 'brain fog', 'trouble concentrating',
'forgetfulness', 'sadness',

   # Sleep-related issues
   'insomnia', 'trouble sleeping', 'difficulty sleeping', 'cant sleep', 'restless sleep',
'waking up at night',
   'nightmares', 'vivid dreams', 'sleep disturbance', 'oversleeping', 'sleep too much',
'sleep paralysis',

   # Skin-related issues
   'rash', 'hives', 'itch', 'itching', 'redness', 'dry skin', 'flaky skin', 'acne',
'breakouts',
   'eczema', 'swelling', 'inflammation', 'puffy', 'puffiness', 'skin irritation',
'bruising',

   # Weight and appetite
   'weight gain', 'weight loss', 'loss of appetite', 'increased appetite', 'overeating',
'cravings',
   'binge eating', 'bloating', 'water retention', 'swelling in legs', 'fluid retention',

   # Cardiovascular-related
   'palpitations', 'heart racing', 'fast heartbeat', 'slow heartbeat', 'chest pain', 'chest
tightness',
   'high blood pressure', 'low blood pressure', 'hypertension', 'fainting', 'dizziness
standing up',

   # Vision and eyes
   'blurry vision', 'vision changes', 'eye pain', 'dry eyes', 'watery eyes', 'sensitive to
light', 'double vision',

   # Mouth, throat, and breathing
   'dry mouth', 'cotton mouth', 'thirsty', 'sore throat', 'cough', 'difficulty breathing',
'shortness of breath',
```

```
    'wheezing', 'congestion', 'runny nose', 'stuffed nose', 'nosebleeds', 'difficulty
swallowing', 'hoarse voice',

    # Sexual and reproductive system
    'loss of libido', 'low sex drive', 'erectile dysfunction', 'impotence', 'painful
intercourse', 'vaginal dryness',
    'menstrual changes', 'irregular periods', 'missed period', 'spotting', 'cramps', 'heavy
periods', 'prolonged periods',
    'discharge', 'brown discharge', 'breast tenderness', 'breast pain', 'nipple soreness',

    # Allergy-related
    'allergic reaction', 'swelling of the face', 'swelling of lips', 'swelling of tongue',
'difficulty breathing',
    'hives', 'itching', 'anaphylaxis', 'rash', 'throat swelling',

    # Neurological and cognitive
    'tremors', 'shaking', 'numbness', 'tingling', 'pins and needles', 'seizures',
'convulsions', 'loss of balance',
    'dizziness', 'vertigo', 'coordination issues', 'memory loss', 'confusion',

    # Urinary and renal system
    'frequent urination', 'urgent urination', 'difficulty urinating', 'painful urination',
'kidney pain', 'dark urine',
    'blood in urine', 'urinary tract infection', 'UTI', 'incontinence', 'trouble holding
urine',

    # Miscellaneous
    'fever', 'chills', 'sweating', 'night sweats', 'hair loss', 'brittle nails', 'ringing in
the ears', 'tinnitus',
    'bad taste', 'metallic taste', 'dry lips', 'swollen lips', 'joint stiffness'
]

# Common negation words or phrases
negation_words = [
    # Single-word negations
    "no", "not", "never", "none", "nothing", "nowhere", "neither", "nobody", "without",
"hardly", "barely", "scarcely",

    # Common contractions
    "dont", "doesnt", "didnt", "wont", "wouldnt", "isnt", "arent", "wasnt", "werent",
"havent", "hasnt",
    "hadnt", "cant", "couldnt", "shouldnt", "mustnt", "mightnt", "shant", "aint",

    # Colloquial and informal negations
    "aint", "nah", "nope", "nothin", "not a single", "not any", "didnt", "doesnt", "havent",
"isnt", "arent", "wasnt", "werent",

    # Phrases indicating negation
    "by no means", "in no way", "on no account", "under no circumstances", "neither nor", "no
longer", "no way", "not at all",
    "not even", "not once", "not anymore", "far from", "seldom", "few if any", "least of all"
]
```

Appendix D: Parser

```
import re
import urllib.parse
import html

def parse_string(input):
   # Decode HTML entities and URL-encoded characters in a single pass
   input = html.unescape(urllib.parse.unquote(input))

   # remove excess white space and convert hyphens to spaces
   input = re.sub(r'[\-]+|\s+', ' ', input).strip()

   # remove any special characters
   input = re.sub(r'[^a-zA-Z0-9 ]+', '', input)

   return input.lower()

def find_first_occurence(key, arr):
   if key == '':
       return -1

   for (i, item) in enumerate(arr):
       # if the item is 3 characters or less, skip.
       # else if there is a match found
       if len(item) <= 3:
           continue
       elif item in key or key in item:
           return i

   return -1
```